Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Mejoras en una herramienta para la gamificación

TITULACIÓN: Grado en Ingeniería de Sistemas Aeroespaciales

AUTORES: Albert Lillo Méndez
Pol Peinado Alcaide
Sergio Sánchez Plaza

DIRECTORES: Miguel Valero
Roc Meseguer

FECHA: 29 de agosto del 2019

Título: Mejoras en una herramienta para la gamificación

Autores: Albert Lillo Méndez
Pol Peinado Alcaide
Sergio Sánchez Plaza

Directores: Miguel Valero
Roc Meseguer

Fecha: 29 de agosto del 2019

Resumen

El proyecto descrito en este documento se ha centrado en la reestructuración y mejora de ClassPip, que es una aplicación para crear escenarios de gamificación en entornos educativos. ClassPip es el resultado del trabajo colaborativo de varios alumnos de la EETAC desde hace dos años y nuestro TFG representa, por tanto, una contribución más a este proyecto.

La gamificación es un recurso de aprendizaje utilizado en ámbitos no lúdicos, que permite trasladar las mecánicas, dinámicas y estéticas de los juegos con la finalidad de optimizar los resultados académicos-profesionales, interiorizar conocimientos con métodos más entretenidos y generar experiencias positivas a los usuarios consiguiendo despertar la motivación, compromiso y ánimo de superación.

El objetivo de este proyecto es rehacer la versión anterior mejorando las funcionalidades, la navegabilidad interfaz gráfica y, añadir nuevas funcionalidades. Esto incluía la realización de una nueva base de datos y aplicación web. Este proyecto surge una vez analizada la versión anterior, la cual mostraba ciertas carencias en el ámbito de organización y la imposibilidad de realizar cambios de forma sencilla al crear nuevas versiones. Por lo tanto, uno de los objetivos principales más importantes era conseguir un código entendible y más fácil de evolucionar, el cual se ha conseguido mediante una reorganización completa del código y el uso de ficheros de texto que explican todas las funciones del código de cada componente. Además, la versión anterior muestra carencias en la navegabilidad e incoherencias en el estilo gráfico. Por tanto, la reorganización realizada a mejorado notablemente la navegabilidad y ha implementado una estrategia para centralizar el estilo grafico de forma que el estilo sea coherente y fácil de alterar.

Otro de los objetivos era crear una aplicación móvil desde cero, y que debía diferenciarse con la modalidad web. Esto se ha conseguido reduciendo el número de funcionalidades respecto a la aplicación web, nuevo diseño funcional se han realizado tanto en la aplicación web como en mobile.

Tanto la base de datos como la modalidad web se ha desarrollado con Angular, manteniendo las herramientas de las versiones anteriores y la aplicación móvil ha sido desarrollada mediante Ionic.

Se ha utilizado la herramienta Git para poder realizar versiones del proyecto de forma paralela, pudiéndolas mezclar de forma sencilla y minimizando los errores.

Este documente se centra en explicar cómo se ha restructurado todo el proyecto Classpip. Empezando por el desarrollo de una nueva base de datos. Además, se reorganizan las funcionalidades tanto de la modalidad web como la del móvil creando nuevos diseños funcionales. Por último, se muestra una comparativa entre la versión anterior y la versión actual, donde se aprecian los grandes cambios realizados en este proyecto.

Title: Improvement of a gamification tool functionalities

Authors: Albert Lillo Méndez
Pol Peinado Alcaide
Sergio Sánchez Plaza

Directors: Miguel Valero
Roc Meseguer

Date: August 29 th 2019

Overview

The project described in this document has focused on the restructuring and improvement of ClassPip, which is an application for creating gamification scenarios in educational environments. ClassPip is the result of the collaborative work of several students of the EETAC since two years ago and our TFG represents, therefore, one more contribution to this project.

Gamification is a learning resource used in non-ludic fields, which allows to transfer the mechanics, dynamics and aesthetics of the games in order to optimize the academic-professional results, internalize knowledge with more entertaining methods and generate positive experiences to the users managing to arouse the motivation, commitment and spirit of overcoming.

The aim of this project is to remake the previous version by improving the functionalities and the navigability of the graphical interface. That included the creation of a new database and web application. This project arose analyzing the previous version, which showed certain shortcomings in the area of organization and making changes easily in new versions. Therefore, one of the most important objectives was to achieve an understandable code and easier to evolve, which has been achieved through a complete reorganization of the code and the use of text files that explain all the functions of the code of each component. In addition, the previous version shows shortcomings in navigability and inconsistencies in graphic style. Therefore, the reorganization carried out has notably improved the navigability and has implemented a strategy to centralize the graphic style, in such a way that the style is coherent and easy to alter.

Another objective was to create a mobile application from scratch, which had to be differentiated with the web mode. This has been achieved by reducing the number of functionalities with respect the web application, new functional design have been made for both, in the web application and in the mobile application.

The database and the web modality have been developed with Angular, maintaining the tools of previous versions and the mobile application has been developed using Ionic.

Git tool has been used to make parallel versions of the project, being able to mix them in a simple way and minimizing errors.

This document focuses on explaining how the Classpip project has been restructured. Starting with the development of a new database. In addition, the functionalities of both, the web and the mobile modality, are reorganized, creating new functional designs. Finally, it shows a comparison between the previous version and the current version.

ÍNDICE

INTRODUCCIÓN

En los últimos años, el número de usuarios adictos a tecnologías móviles y videojuegos ha visto un crecimiento significativo. Del mismo modo, los sistemas de enseñanza tradicionales parecen no despertar interés en los nuevos alumnos y dificultan la interacción alumno-profesor, por ello es necesaria la creación de un nuevo entorno educativo utilizando mecánicas y dinámicas de videojuegos donde consigamos despertar el interés del usuario y así responder con un incremento del rendimiento académico. Con tal de ofrecer una solución a la demanda de interés educativo, se realizó el proyecto del aplicativo Classpip.

Classpip no es más que un entorno gamificado que hace uso de dinámicas, mecánicas y estéticas de videojuegos. Dicho aplicativo es un proyecto desarrollado por distintos alumnos de la EETAC de diversas generaciones. Este está compuesto por distintos módulos que generalmente suelen ser desarrollados de manera autónoma por cada alumno.

Para el caso de nuestro grupo, nos enfrentamos a la creación de un aplicativo Classpip desde el cero, ya que se realizó la consideración de que los proyectos anteriores realizaron un enfoque a nivel Instituto de la aplicación dificultando la comprensión de la misma y haciendo el código más sensible a errores. Esto implicó enfrentarnos desde el origen a la creación de una nueva base de datos, a un nuevo Dashboard y un nuevo Mobile.

La estructura del trabajo se puede resumir de la siguiente manera:

Introducción sobre que es la gamificación y como se aplica en centros educativos, recogida en el **Capítulo 1**. Estos ejemplos ayudan a entender cómo funciona la gamificación a la par que proporcionan ideas para mejorar el proyecto Classpip.

El proyecto Classpip viene definido en el **Capítulo 2**, donde se profundiza en su arquitectura y sus características.

En el **Capítulo 3** se explica detalladamente el planteamiento inicial y la resolución final que se enfocó para el proyecto. El objetivo principal que nos planteamos en este proyecto es el generar un código simple consiguiendo implementar todas las funcionalidades solicitadas, a la vez que buscar variables que sean claras y funciones que al inicializarte en la aplicación puedas entender en un intervalo de tiempo reducido, para ello se ha realizado el esfuerzo de comentar todos los archivos de la aplicación y estandarizar los estilos gráficos para que los próximos compañeros tengas un patrón al cual regirse.

El diseño funcional, reestructurando la aplicación para cada uno de los proyectos se muestra en el **Capítulo 4**. Estos diseños son validados por los

directores del proyecto para que los futuros alumnos que se encarguen de este proyecto puedan seguir sin necesidad de volver a empezar de nuevo.

El estudio de prototipado de pantallas para Dashboard y Mobile se encuentra en el **Capítulo 5**. Estos diseños estéticos fueron previamente valorados por el grupo y directores del proyecto para así conseguir un patrón al cual regirse en las próximas versiones de Classpip.

Al iniciarnos con Classpip, observamos que el planteamiento de la base de datos no era el esperado, ya que la base de datos heredada estaba basada en una aplicación a nivel Colegio, cuando el planteamiento actual iba a un nivel docente. Consecuentemente, esto implicó la creación de una nueva base de datos y sus consecuentes de modelos y relaciones de modelos. En el **Capítulo 6** se explica detalladamente el proceso seguido y que relaciones se han utilizado para definir la base de datos actual.

En el **Capítulo 7** se explica que estilos gráficos han sido utilizados para cada estilo de pantalla, es decir, se definen los elementos que deben aparecer en las pantallas de tipo creación, edición, etc. Se recogerán en el capítulo tantos los estilos gráficos de Dashboard como de Mobile Profesor.

Una comparativa entre la versión anterior y la actual para poder diferenciar los cambios que se han realizado en el aspecto gráfico y organizativo de Classpip se muestra en el **Capítulo 8**.

Para conocer la cantidad de archivos generados, clases, servicios, componentes, etc. Se ha generado el **Capítulo 9,** donde se recoge toda la información referente a la cantidad de código existente en Dashboard y Mobile Profesor.

Una vez conocida la magnitud del proyecto, se realiza un conjunto de pruebas y evaluaciones a diversos docentes de distintas edades y distintos países para conocer cuánto robusto es el aplicativo y que amigable y sencillo es para el usuario. Los resultados se encuentran en el **Capítulo 10** de la memoria.

En el **Capítulo 11** se definen las conclusiones técnicas, personales obtenidas durante la realización del proyecto además de las posibles futuras mejoras para desarrolladores del aplicativo Classpip.

El último capítulo de la memoria hace referencia a los anexos que recogen un manual sobre cómo se debe realizar la instalación del aplicativo Dashboard, Mobile y su correspondiente Services, además de los modelos que han sido utilizados en la base de datos con sus correspondientes variables y relaciones.

Durante el desarrollo del proyecto los integrantes del trabajo hemos crecido profesionalmente. Desde el principio tuvimos que organizarnos para repartirnos las tareas y resolver los problemas que nos íbamos encontrando. El trabajo en equipo ha sido la base de este proyecto, si uno de nosotros no sabía realizar una tarea y se estancaba, había otro integrante dispuesto a ayudar y sacarlo a delante. Una de las fases que más se ha implicado el contexto ingenieril ha

sido la de buscar en Internet como resolver nuestros problemas, hemos sido capaces de reutilizar componentes que otros desarrolladores habían realizado y compartido con la comunidad, esto hace convertirnos en mejores ingenieros ya que lo más importante en un ingeniero no es crear algo sino utilizar lo existente y las aplicarlo a la situación oportuna o modificarlo para adaptarlo y resolver el problema.

Como ingenieros todo lo explicados anteriormente nos a porta un valor inconmensurable, algo que probablemente no hubiéramos conseguido realizando otro proyecto o realizándolo individualmente. Por lo que ninguno de los integrantes se arrepiente de elegir este proyecto.

Ahora es el momento de comenzar a explicar el concepto de gamificación.

CAPÍTULO 1. GAMIFICACIÓN

En el siguiente capítulo se realiza una introducción en el concepto de Gamificación, qué técnicas se deben aplicar para gamificar un entorno además del estudio de 3 casos de entornos gamificados en ámbitos educativos.

1.1. ¿Qué es la Gamificación?

La Gamificación es un recurso de aprendizaje utilizado en ámbitos no lúdicos, que permite trasladar las mecánicas, dinámicas y estéticas de los juegos con la finalidad de optimizar los resultados académicos-profesionales, interiorizar conocimientos con métodos más amenos y generar experiencias positivas a los usuarios consiguiendo despertar la motivación, compromiso y ánimo de superación.

Un entorno gamificado utiliza técnicas mecánicas, dinámicas y estéticas extraídas de los juegos. A continuación, se definirán los conceptos básicos de dichas técnicas:

La **técnica mecánica** es definida como la manera de recompensar al usuario en función de los objetivos enmendados. En la **Figura 1.1** se muestran algunas de las técnicas mecánicas más empleadas.

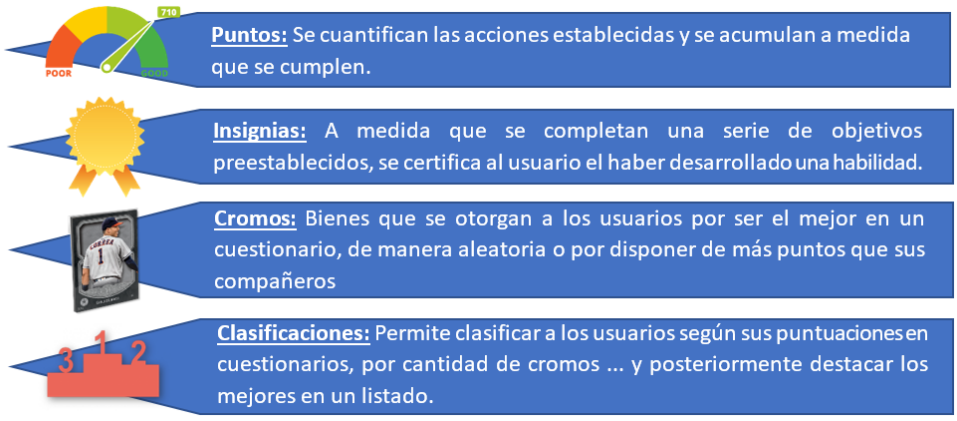


Figura 1.1 Ejemplo de técnicas mecánicas

La **técnica dinámica** es definida como la motivación despertada en el propio usuario para hacerle partícipe y finalmente conquistar los objetivos encomendados. Estas deben ser claras y tienen como objetivo que el usuario tenga una participación directa y continua. En la **Figura 1.2** se muestran algunas de las técnicas dinámicas más empleadas.

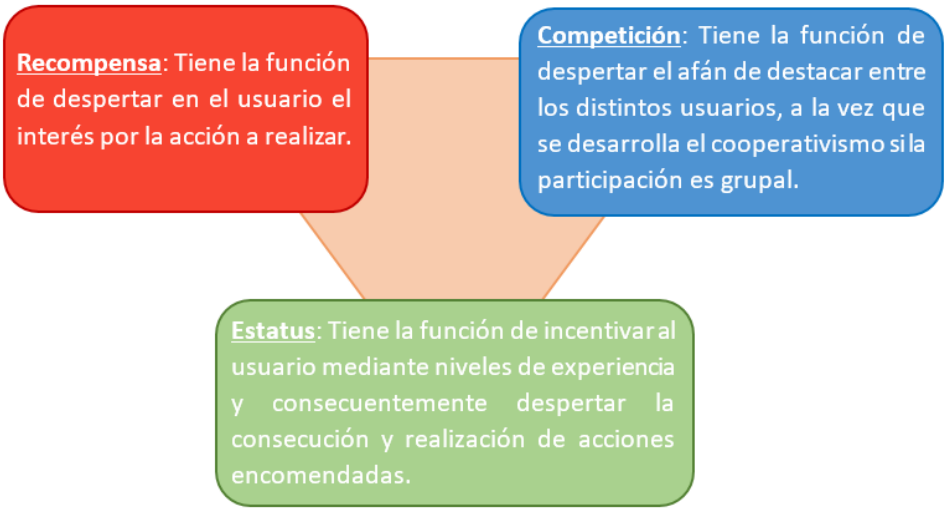


Figura 2.2 Ejemplo de técnicas dinámicas

1.2. Ejemplos de entornos gamificados

Como se ha explicado en el apartado previo, se entiende por gamificación, la técnica de aprendizaje empleada en el ámbito educativo-profesional utilizando mecánicas de juegos para conseguir incrementar el rendimiento académico - profesional a la par de potenciar la creatividad de los alumnos o empleados.

A continuación, se detallan tres ejemplos sobre entornos gamificados y la repercusión que han tenido en el rendimiento académico y motivacional.

1.2.1. Gamificación en el Máster de Educación Especial de la Facultad de Educación de la Universidad Complutense de Madrid

¿De quién estamos hablando y donde ha aplicado la gamificación?

- María L. Calatayud, profesora del máster de Educación Especial de la Universidad Complutense de Madrid aplicó la gamificación en el curso de Intervención Psicológica en Trastornos de Aprendizaje.

¿Por qué un entorno gamificado?

- Tras asistir a un congreso en que los alumnos eran los protagonistas, se replanteó el sistema de enseñanza en sus clases y aplicó la gamificación en su curso.

¿Qué entorno gamificado utiliza?

- El entorno gamificado utilizado es “Los Sabios de Túnica Color Ciruela”. Compaginó el entorno con Kahoot y Aurasma para potenciar la realidad aumentada, Edraws y Bubbles para facilitar la creación de mapas conceptuales y Timeline para generar líneas de tiempo. Además, hizo uso de Telegram para mantener una conexión directa con sus alumnos.

¿Cómo está formado el entorno? ¿Qué dinámicas aplica? (1)

- El entorno gamificado está formado por distintas misiones que permitirán convertir al usuario en un Sabio de Túnica de Color Ciruela.

  Para ello, los aspirantes deberán pasar por un duro y difícil camino basado en misiones.

  Previamente deberán haber entregado el “Compromiso del Aprendiz” y demostrar que son unos aspirantes con curiosidad, valentía y con disposición de ampliar su zona de confort.

  Inicialmente deberán formar grupos y asignar roles a cada participante, desde Investigador a Mentor, definido a través de los Puzles de Aronson (métodos de aprendizaje cooperativo).

Figura 1.3 Roles disponibles en “Los Sabios de Túnica Color Ciruela”

¿Cómo está formado el entorno? ¿Qué dinámicas aplica? (2)

- Cada aspirante dispondrá de puntos de experiencia que le permitirá ir evolucionando desde Shinshei-sha (Nivel de Iniciación) a Sensei (Nivel de Consciencia plena).

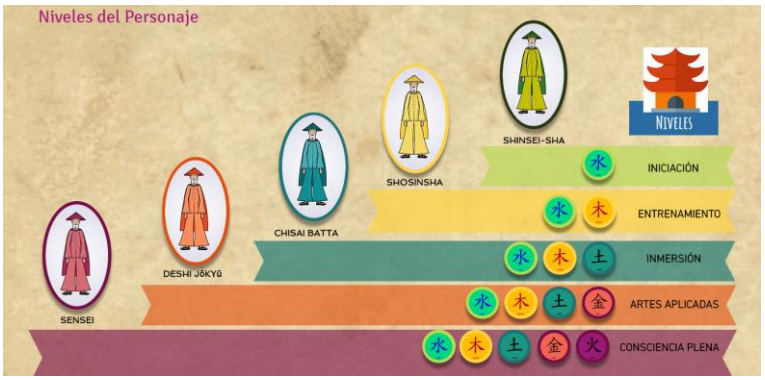


Figura 1.4 Niveles de experiencia disponibles en “Los Sabios de Túnica Color Ciruela”

Durante las misiones, los aspirantes podrán ganar puntos e insignias sobre Bienestar, Experiencia o sabiduría. Además, el entorno dispone de un factor de poderes sorpresa que te permitirán tener una ayuda para superar las misiones establecidas.

Figura 1.5 Puntos e Insignias disponibles en “Los Sabios de Túnica Color Ciruela”

¿Contiene misiones? En caso afirmativo explica brevemente una de ellas.

- Dispone de 5 misiones, que una vez completadas, permiten acceder al Gran Desafío final. Cada misión está formada por unos retos que deberán superarse para poder completar la misión.

  Cojamos el ejemplo de la Misión 1 “Agua para Arcilla “. Para poder completar esta misión se deberá haber demostrado que se dispone del conocimiento los retos del mayor de los tesoros, del poder de las palabras, del poder de la empatía, del poder del pensamiento y de la lucha de Titanes. Una vez demostrado el conocimiento, habrá permiso de acceder a la siguiente misión.

Figura 1.6 Misiones disponibles en “Los Sabios de Túnica Color Ciruela”

¿Existe el factor sorpresa?

- El factor sorpresa está presente en el entorno aportando poderes a los aspirantes y facilitándoles las misiones.

Figura 1.7 Factor Sorpresa en “Los Sabios de Túnica Color Ciruela”

¿Qué privilegios o premios puede conseguir el alumno?

- Pueden conseguir mediante los poderes, privilegios para superar las misiones o a medida que superar misiones, su nivel de sabiduría, bienestar o experiencia aumenta.

¿Qué resultados se han obtenido aplicando la gamificación?

- Se consigue suprimir la connotación negativa hacia los sistemas de evaluación y se mejora el rendimiento académico y creativo de los alumnos.

1.2.2. Gamificación en las clases de Ciencia de la Naturaleza en el IES Antonio de Nebrija de Móstoles

¿De quién estamos hablando y donde ha aplicado la gamificación?

- Javier Espinosa, profesor de Ciencias de la Naturaleza del IES Antonio de Nebrija de Móstoles diseñó un entorno gamificado para los cursos de 1º y 2º siguiendo el ejemplo de videojuego de Paul Andersen (diseñador de videojuegos para aulas).

¿Por qué un entorno gamificado?

- Tras evidenciar que sus alumnos no disfrutaban de la estancia en clase y mediante la charla de TED donde Sir Ken Robinson explicaba cómo "las escuelas matan la creatividad", diseñó un videojuego para los alumnos del curso de Ciencias de la Naturaleza de 1º y 2º.

¿Qué entorno gamificado utiliza?

- El entorno gamificado utilizado es "*Earthxodus*". Hizo uso del constructor de sitios webs Weebly para la creación del entorno.

¿Cómo está formado el entorno? ¿Qué dinámicas aplica?

- El entorno gamificado está posicionado en el año 2080, donde la humanidad debe tomar la difícil decisión de abandonar el planeta tierra a causa de la alta contaminación expuesta.

  Para ello los alumnos, deberán realizar misiones y así consecuentemente recuperar el estado normalizado de la atmosfera, hidrosfera, geosfera y biosfera.

  Esta aventura no la deberán afrontar individualmente, ya que las misiones se realizarán desde unas naves espaciales formadas por 5 científicos cada uno con su correspondiente robot.

  Cada científico deberá adquirir los máximos puntos de experiencia posibles mediante una profunda búsqueda sobre las posibles causas de deterioro del planeta tierra. Contra mayor es el número de puntos de experiencia, mayor será el número de robots que se dispondrán para ayudarlos en la misión.

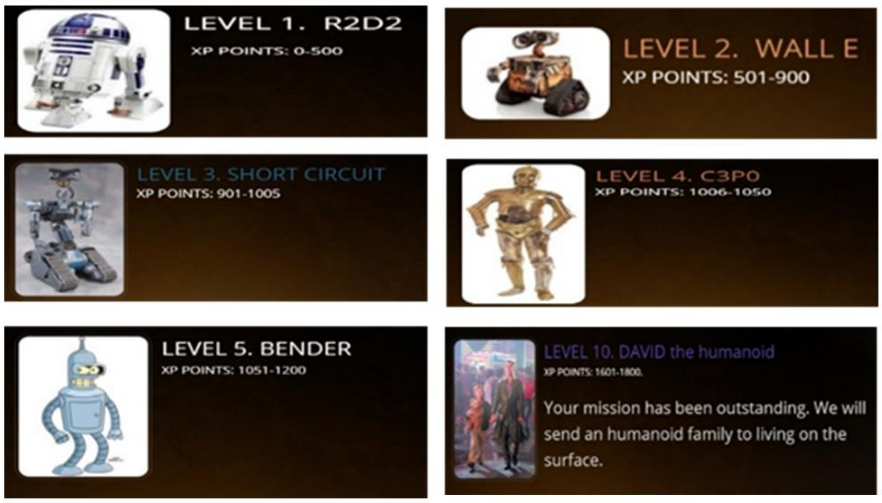


Figura 1.8 Niveles disponibles de “Earthxodus”

¿Contiene misiones? En caso afirmativo explica brevemente una de ellas.

- Dispone de 4 misiones, que una vez completadas, permitirán que la humanidad pueda volver a vivir en el planeta Tierra. Cada misión está enfocada en la búsqueda de las razones de destrucción de las capas y conocer la composición química para analizar si disponen de oxígeno suficiente para abastecer a toda la humanidad.

¿Existe el factor sorpresa?

- El factor sorpresa no se encuentra presente en el entorno gamificado.

¿Qué privilegios o premios puede conseguir el alumno?

- A medida que se obtiene mayor número de puntos de experiencia, el alumno se ve premiado con la ayuda de nuevos robots en la misión espacial.

¿Qué resultados se han obtenido aplicando la gamificación?

- Mediante la gamificación, Javier Espinosa ha conseguido alzar la nota media de sus clases y potenciar la interacción con sus alumnos. No solo se centra en aspectos evaluativos, sino que además consigue explorar las inquietudes y motivaciones de sus alumnos.

1.2.3. Gamificación en las clases de Ciencia Sociales en el IES Torrent de les Bruixes

¿De quién estamos hablando y donde ha aplicado la gamificación?

- Ignacio Maté Puig, profesor de Ciencias Sociales del IES Torrent de les Bruixes, tras un largo periodo en búsqueda de la creatividad en la enseñanza con fin de generar experiencias y emociones con sus alumnos, diseñó un entorno gamificado basado en la serie de RTVE "Ministerio del tiempo".

¿Por qué un entorno gamificado?

- El "Ministerio del tiempo" despertó en Ignacio, la oportunidad de darle un punto de vista atractivo a la historia mediante un entorno gamificado, convirtiendo a sus alumnos en agentes que deberán viajar en el tiempo para evitar que la historia cambie.

¿Qué entorno gamificado utiliza?

- El entorno gamificado utilizado es "*El ministeri del temps"*. Hizo uso del constructor de sitios webs Weebly para la creación del entorno.

¿Cómo está formado el entorno? ¿Qué dinámicas aplica?

- El entorno gamificado está formado por distintas misiones que evitarán que la Historia sea modificada y consecuentemente no afecte a la vida actual. Para ello, los alumnos se convierten en funcionarios del ministerio que deberán retroceder en el tiempo y evitar que la historia se modifique.

  Para ello, inicialmente los aspirantes deberán redactar una ficha técnica en la primera misión, para demostrar que son capaces de seguir las instrucciones y consecuentemente poder retroceder en el tiempo.

  Posteriormente, los funcionarios deberán escoger en que especialidad trabajarán. Deberán escoger entre agente de campo o agente de investigación, cada especialidad tendrá unos objetos que le ayudarán en las misiones correspondientes.

  Además, cada alumno dispondrá de puntos de experiencia, adquiridos a medida que se completan las misiones asignadas. Estos puntos les permitirán ir progresando dentro del ministerio, pasando de ser un ayudante de becario no contratado a ministro.

¿Contiene misiones? En caso afirmativo explica brevemente una de ellas.

- Dispone de 4 misiones. Cada misión está enfocada en una misión de viaje en el tiempo.

Cojamos el ejemplo de la Misión 2, donde retrocedemos al año 755 a la edad Media. Los alumnos deberán asegurar la llegada de Abderramán al Ándalus. Para completar la misión, deberán realizar unos vídeos explicativos sobre las características principales de la Edad Media ya que en la Misión 3 ha aparecido un virus en la base de datos y ha borrado toda la información referente a la Edad Media.

¿Existe el factor sorpresa?

- El factor sorpresa no se encuentra presente en el entorno gamificado.

¿Qué privilegios o premios puede conseguir el alumno?

- A medida que se obtiene mayor número de puntos de experiencia, el alumno se ve premiado con recompensas como puede ser beneficios individuales en pruebas calificativas, eliminar preguntas de controles o entregar trabajos un día posterior a la fecha establecida.

¿Qué resultados se han obtenido aplicando la gamificación?

- Ignacio Maté, mediante la gamificación ha conseguido despertar interés en sus alumnos por la historia, mejorar el rendimiento educativo y contagiar a otros profesores mediante congresos, que la gamificación es la nueva vía para crear un entorno educativo creativo y en el cual los estudiantes son los protagonistas.

CAPÍTULO 2. CLASSPIP

Classpip es una herramienta de Gamificación desarrollada por estudiantes de la Escuela de Ingeniería de Telecomunicaciones y Aeroespacial de Castelldefels, perteneciente a la Universidad Politécnica de Cataluña. Mediante esta herramienta, se pretende fomentar la motivación e interés de los alumnos en sus estudios mediante la utilización de dinámicas/mecánicas de juegos y las nuevas tecnologías. A diferencia de los ejemplos anteriores, Classpip es más versátil ya que permite configurar las mecánicas y dinámicas según el deseo del docente para cada clase, en cambio los ejemplos previos eran dinámicas y mecánicas para un entorno definido.

La aplicación se inició en el 2016 por el Departamento de Arquitectura de Computadores como un proyecto de final de grado de un alumno. Aunque inicialmente se centró en la arquitectura, posteriormente se han desarrollado una serie de módulos independientes llevados a cabo por alumnos de distintas promociones. Estos módulos fueron agrupados en el proyecto previo, permitiendo así la interacción entre ellos.

La implicación de varios alumnos año tras año en el proyecto ha permitido un crecimiento significativo en estos años. Actualmente Classpip dispone de varias funcionalidades y juegos disponibles, además de una aplicación de Mobile para el profesor.

2.1. Arquitectura del proyecto

Para el desarrollo de una herramienta como Classpip es sumamente importante elegir las tecnologías adecuadas que permitan mostrar la información correctamente e interconectar los diferentes módulos. Del mismo modo, la forma en que se muestran y se tratan los datos serán de suma importancia y deberán regirse por el patrón Modelo – Vista – Controlador (MVC).

Actualmente Classpip dispone de una aplicación web para el profesor (Dashboard) y una aplicación móvil del profesor para plataforma iOS (Mobile Profesor). Además, se ha realizado el diseño funcional y estético de las pantallas de la aplicación móvil para el alumno (Mobile Alumno), con su correspondiente base de datos API REST.

Se ha hecho uso de las tecnologías empleadas en los proyectos previos. Estas son:

- **Strongloop LoopBack 3:** permite gestionar la API REST (generando modelos, relaciones, …).

- **Visual Studio Code:** editor de código.

- **Angular Material:** mediante sus componentes se consigue una interfaz gráfica amigable y atractiva.

- **Bootstrap:** funcionalidad similar a la de angular material.

- **Cordova/ionic:** permiten desarrollar la interfaz gráfica del aplicativo Mobile.

- **GitHub:** al realizar un proyecto cooperativo como este, es de suma importancia compartir de manera cómoda y sencilla las contribuciones de cada miembro del equipo. GitHub permite guardar tu proyecto en la nube y juntar el progreso de cada alumno. La explicación sobre cómo compartir el progreso, se encuentra en uno de los videos tutoriales de las Lecciones.

2.2. Descripción de la aplicación

Classpip es una herramienta creada para el profesor en la que podrá inscribir y gestionar los grupos a los que imparte clase, creando juegos individuales y por equipos. Estos equipos podrán ser confeccionados por el profesor.

Actualmente disponemos de tres tipos de juegos: por puntos, de colección y de competición. Paralelamente a este proyecto se está desarrollando el juego de cuestionario.

El profesor podrá personalizar los juegos mediante la creación de los materiales para jugarlos. Esto es:

- Creando los puntos e insignias que el profesor podrá otorgar a los alumnos en un juego de puntos.

- Creando las colecciones y sus respectivos cromos que podrá repartir el profesor a los alumnos en los juegos de colección.

Adicionalmente, se podrá usar la aplicación para pasar lista y controlar la asistencia de los alumnos y premiar la puntualidad.

1.2.1. Juego de puntos

La modalidad “puntos” consiste en repartir puntos a los alumnos/equipos premiando ciertas actitudes y logros (comportamiento, puntualidad, buena nota en el examen, …). Este juego dispondrá de niveles creados por el profesor en

el momento de generar el juego. Los niveles otorgarán un privilegio determinado a los participantes que lo alcancen.
El objetivo es ser el primero en la clasificación e ir alcanzando estos niveles para ir obteniendo las recompensas correspondientes.

Por ejemplo, el profesor crea un juego de puntos en el que escoge jugar con los tipos de puntos mencionados anteriormente; comportamiento, puntualidad y buena nota en el examen. El juego, además, contará con tres niveles a alcanzar con sus respectivos privilegios: bronce, plata y oro.

- Bronce: pueden salir 5 minutos antes al patio. Se consigue llegando a los 10 puntos.
- Plata: pueden entregar unos deberes un día más tarde. Se consigue llegando a los 25 puntos.
- Oro: pueden repetir una pregunta de un examen en la que hayan fallado. Se consigue llegando a los 50 puntos.

El profesor asignará puntos especificando el tipo de punto que asigna y el valor que le da. Por ejemplo, podríamos escoger asignar 1 punto por buen comportamiento a Sergio o 2 puntos por buena nota en el examen a Pol.

1.2.2. Juego de colección

La modalidad “colección” consiste en premiar a los alumnos/equipos con cromos pertenecientes a la colección previamente creada y escogida para el juego. Estos cromos los puede asignar el profesor de manera deliberada o de forma aleatoria. El objetivo de los participantes es obtener el álbum completo.

Por ejemplo, el profesor crea un juego de colección que se va a jugar con la colección llamada NBA, la cual cuenta con cromos de diferentes jugadores.

Cada cromo tiene una probabilidad de aparición en sobres aleatorios en función de su nivel. Por consiguiente, el profesor podría poner como recompensa por alcanzar objetivos algún cromo determinado. Por ejemplo, todo aquel que saque más de un 9 en el examen de un día concreto obtendrá el cromo de LeBron James.

Otra opción es poner como premio un sobre de cromos aleatorios. Siguiendo con el ejemplo anterior, el premio por sacar más de un 9 podría ser un sobre con 4 cromos.

A diferencia del juego de puntos, no se especifican unos objetivos marcados cuando creas el juego para poder obtener recompensas.

1.2.3. Juego de competición

En la modalidad “competición”, cuando creamos un juego, podemos elegir entre tipo “Liga” y tipo “Torneo”. En función de la opción que se escoja, el objetivo será uno u otro.

- **Liga**: el objetivo es quedar lo más alto posible en la clasificación. Cada jornada habrá un enfrentamiento entre dos participantes. El ganador de la jornada se decidirá en base a un criterio decidido por el profesor para esa fecha concreta. El ganador obtendrá puntos para la clasificación en la liga y, además, podría recibir otro tipo de premios para otros juegos en activo (como por ejemplo cromos).

  Por ejemplo, pongamos que en la clase tenemos 10 alumnos. Eso supone 18 jornadas. Podremos escoger la fecha de cada jornada, que serán los días que tengamos clase con el profesor que ha creado el juego. El profesor decide que para la jornada 1, el ganador será el que haya sacado más nota en el examen anterior. Para la jornada 2, sin embargo, decide que el ganador será el que tenga mayor calificación en un trabajo y, además de los puntos en la clasificación de la liga, obtendrá 3 cromos aleatorios. Se deberá dejar claro el criterio ganador de cada jornada y las respectivas recompensas.

- **Torneo**: se jugarán partidos eliminatorios hasta que quede un ganador. Al igual que en el modo “Liga”, se deberán especificar los criterios ganadores de cada encuentro y sus respectivas recompensas. La diferencia más remarcable con la otra modalidad es que, una vez que quedas eliminado, ya no puedes optar a más recompensas.

  Utilizando los criterios anteriores, el ganador de un partido podría ser el que mayor nota obtenga en un examen y, su recompensa, pasar de ronda y 3 cromos aleatorios.

CAPÍTULO 3. OBJETIVOS Y PLAN DE TRABAJO

Al aceptar este proyecto nos marcamos una serie de objetivos que cambiaron completamente al afrontarlo. Inicialmente, la idea de este trabajo era reestructurar el proyecto Dashboard (añadiendo algunas funcionalidades nuevas), el Services y la página de Onboarding. Por otro lado, empezaríamos de cero los dos proyectos Mobile.

No obstante, una vez teníamos el diseño funcional completado, verificamos el código heredado y nos pareció que, pese al esfuerzo por organizar y simplificar el código por compañeros en proyectos anteriores, éste continuaba siendo muy complejo. Los nombres de los componentes, las funciones y los servicios eran poco esclarecedores y dificultaban la compresión del código.

Cabe destacar que el hecho de haber hecho módulos independientes y posteriormente juntarlos, limitaba mucho las funcionalidades del Dashboard y su organización.

Consideramos que iba a ser más complejo comprender el código heredado por los compañeros anteriores y después reestructurarlo todo como habíamos diseñado, que empezarlo de cero manteniendo la misma idea. Esta dificultad no solo se debía al código, sino también a que las relaciones en la API cambiaban completamente.

En proyectos anteriores se había considerado que Classpip iba a ser una herramienta para la escuela. Con lo que se incluía la escuela, todos los profesores, alumnos, cursos, asignaturas, etc. Para simplificar nosotros consideramos que iba a ser una herramienta del profesor. Cada profesor tendrá su usuario sin importar la escuela a la que pertenezca.

Después de debatirlo con los tutores del proyecto, decidimos que la mejor decisión era empezar el proyecto de cero, lo que cambió completamente los objetivos. Éstos pasaron a ser los siguientes:

- Hacer un buen diseño funcional, reestructurando la aplicación para cada uno de los proyectos. Esto diseños serán validados por los directores del proyecto para que los alumnos de los próximos cursos puedan seguir sin necesidad de volver a empezar de nuevo.

- Rehacer el proyecto Dashboard mejorando las funcionalidades e interfaz gráfica y, además, añadiendo funciones nuevas. Pese a que intentaremos hacer la aplicación lo más atractiva visualmente posible, no pretendemos marcar el estilo que va a seguir la aplicación de ahora en adelante, cosa que si pretenderemos en el Mobile.

- Facilitar el uso de la aplicación y hacerla más intuitiva para el usuario.

- Hacer un código más sencillo de entender para los nuevos programadores.

- Rehacer el proyecto Services, eliminando el modelo escuela y basándonos en el modelo profesor.

- Crear la aplicación Mobile para profesor.

Debido a esta modificación de objetivos, la web de Onboarding decidimos no abordarla y así centrarnos en los otros tres proyectos restantes.

Con los objetivos iniciales, nuestro plan de trabajo se repartía de manera que, como teníamos tres proyectos por delante (Dashboard, Mobile Profesor y Mobile Alumno) y nuestro equipo estaba formado por tres personas, cada uno se encargaría de uno. Éste se vio afectado con la modificación de objetivos.

Con los nuevos objetivos en mente, hicimos definimos otro plan de trabajo. Inicialmente, decidimos que Pol se encargaría de iniciar las funcionalidades del Dashboard a la par que generaba las pantallas de manera sencilla. Albert y Sergio se encargarían de definir la estética de estas pantallas. El proyecto Service se iría abordando entre los tres.

No obstante, vimos que esta organización no era adecuada, y Sergio se pasó a realizar el proyecto Mobile Profesor paralelamente a el desarrollo del Dashboard adaptando las funcionalidades a éste.

Por lo tanto, finalmente Pol se encargaría de las funcionalidades del Dashboard y del Services, Albert de la parte gráfica y ayudar a Pol con algunas funcionalidades del Dashboard y Sergio del Mobile Profesor.

CAPÍTULO 4. DISEÑO FUNCIONAL

Como se ha explicado en el apartado de objetivos del trabajo, el hecho de juntar módulos independientes conlleva complejidad y falta de organización.

Por lo tanto, es primordial antes de abordar el código, hacer un buen diseño funcional, reestructurando la aplicación de manera adecuada, ya que se seguirá esa organización en los proyectos venideros. Además, este diseño también determinará la manera en que la API estará organizada y los modelos y relaciones que tendrá.

El diseño funcional se definirá tanto para Dashboard cómo para modalidad Mobile.

4.1. Dashboard

Para visualizar el diseño funcional se hará uso de listas multinivel. Además, este diseño funcional vendrá recogido en la **Figura 4.1** al final del apartado para facilitar su entendimiento.

Hay que tener en cuenta que se dispone de cuatro grandes bloques a partir de los cuales se desarrolla la aplicación: Grupos, Cuestionarios (en desarrollo por otro compañero), Colecciones y Puntos. Estos cuatro bloques se incluirán en el navbar para poder navegar de manera fácil e intuitiva por la aplicación.

4.1.1. Grupos

Permite al profesor acceder a los grupos que ya tiene creados o en su caso, crear nuevos.

Crear grupo: para crear grupo, se deberá definir un nombre y una descripción. A partir de ese momento, el nuevo grupo aparecerá en la lista “mis grupos”. Opcionalmente, se podrán añadir alumnos al grupo cuando se estén creando o añadirlos posteriormente en el modo “editar grupo”.

Mis grupos: el usuario visualizará una lista con los grupos existentes y podrá seleccionar el que desee. Una vez seleccionado el grupo, aparecerá una pantalla que permitirá:

- **Editar grupo**: permite modificar el nombre y/o la descripción. También permite al usuario añadir/eliminar alumnos.

- **Eliminar grupo**: elimina el grupo y las inscripciones de los alumnos en él.
- **Pasar lista**: el usuario podrá pasar lista de manera sencilla mediante el uso de checkboxs en una lista.

- **Equipos**: el usuario visualizará una lista de los equipos creados en el grupo en el que se encuentre o podrá crear nuevos.

    - **Crear equipo**: el usuario definirá un nombre al equipo y podrá ponerle un logo. Posteriormente, al igual que cuando se genera un grupo, opcionalmente podrá añadir los alumnos en el momento de creación del equipo o añadirlos posteriormente en editar equipo.

    - **Lista de equipos**: muestra los equipos que existen en ese momento, de manera que al seleccionar uno se abrirá un desplegable con el logo del equipo (si dispone de logo), los miembros del equipo y la opción de editar el equipo o borrarlo.

    - **Editar equipo**: permite modificar el logo (o añadirlo si no disponía) del equipo, agregar al equipo solo aquellos alumnos que no tienen equipo dentro del grupo, mover alumnos entre equipos del mismo grupo de manera sencilla y eliminar alumnos del equipo.

    - **Eliminar equipo**: elimina el equipo y sus alumnos podrán ser seleccionados para otros equipos del grupo.

Juegos: los juegos es la parte más relevante del proyecto, ya que toda la aplicación gira entorno a la unificación de educación y juegos.

- **Crear juego**: permite crear un juego de los tipos siguientes.

    - **Crear juego de puntos**: para configurar este juego se deben seguir los siguientes pasos.

      - Elegir los tipos de puntos que se van a ser utilizados en el juego, eligiéndolos de todos los puntos que el usuario haya creado (en el apartado *Puntos e Insignias*).

      - Definir el modo de juego (individual o en equipos, si es que se han definido equipos en el grupo).

      - Crear los diferentes niveles, especificando los puntos necesarios para alcanzar dicho nivel y los privilegios que obtienen los alumnos al lograrlo. Opcionalmente el usuario podrá añadir una imagen al nivel.

    - **Crear juego de colección**: para configurar este juego se deben seguir los siguientes pasos.

- Elegir la colección de entre las que el usuario haya creado (en el apartado *Colecciones*).

- Definir el modo de juego (individual o en equipos).

- Definir las reglas de asignación de cromos. Algunas reglas serán manuales mientras que otras pueden ser automatizadas (por ejemplo: se asignará un sobre de 5 cromos cada viernes a las 12:00 a los tres alumnos con más puntos en el juego de puntos en ese momento). Las reglas automáticas deberemos predefinirlas en el programa.

- **Crear juego de competición**: para configurar este juego se deben seguir los siguientes pasos.

- Elegir si la competición va a ser de tipo “Liga” o “Torneo”.

- Definir el modo de juego (individual o en equipos).

- Si es “Liga”, seleccionar el número de jornadas.

- Especificar el criterio que se usará para determinar los ganadores de los emparejamientos de cada jornada.

- Definir las reglas de asignación de premios. Algunos premios se asignarán manualmente y otros de manera automática (por ejemplo: puntos del juego de puntos o sobre de cromos para los 3 primeros de la clasificación después de cada jornada).

- Permite elegir el día en el que se jugará la jornada (por ejemplo: como cada martes hay clase de una asignatura, pongo un partido de liga o uno de torneo cada martes).

- **Crear juego de cuestionario**: funcionalidad desarrollada paralelamente a nuestro proyecto por otro alumno.

- **Geocaching**: funcionalidad desarrollada paralelamente a nuestro proyecto por otro alumno

- **Avatares**: funcionalidad desarrollada paralelamente a nuestro proyecto por otro alumno

- **Listar juegos**: mostrará una lista de los juegos activos e inactivos, permitiendo filtrar la lista por tipo de juego. Al elegir un juego podremos tomar todas las decisiones posibles en relación al juego.

- **Juego de puntos**

  - Asignar cómodamente puntos a los alumnos/equipos (según el modo de juego), identificando el tipo de punto asignado y el valor que el usuario le quiere dar.
  - Mostrar el ranking total y por tipo de punto, identificando los alumnos/equipos que están en cada nivel.

  - Visualizar el progreso del alumno en el juego, indicando su posición, nivel, puntos restantes hasta el siguiente nivel y un historial de puntos (indicando el tipo de punto, el valor y la fecha en que se le asignó).

  - Permite borrar una asignación de puntos de un alumno.

- **Juego de colección**

  - Asignar cómodamente cromos a los alumnos/equipos (según el modo de juego), identificando el cromo asignado.

  - Mostrar una lista de alumnos/equipos y al clicar poder identificar los cromos que dispone.

- **Juego de competición**

  - Asignar premios cómodamente de manera manual, identificando el tipo de punto asignado.

  - Mostrar clasificación de la liga en caso de jugar una competición tipo liga.

  - Mostrar el cuadro de clasificación en caso de jugar una competición tipo torneo de tenis.

  - Mostrar los resultados de los partidos, tanto si es modo liga como torneo de tenis.

  - Decidir los ganadores de los enfrentamientos de cada jornada. El profesor debería ir a la jornada en cuestión y encontrar las opciones para decidir, entre las cuales habría al menos:

    - Manualmente (debe ser cómodo señalar el ganador de cada emparejamiento).

    - Aleatoriamente.

    - El jugador que dispone de más puntos en el juego de puntos (si es que hay juego de puntos).

- El jugador que obtuvo mejor puntuación en un juego de cuestionarios ya terminado (debería poder elegir el profesor el juego de cuestionarios que debe usarse).

- El jugador que disponga de más cromos de la colección de cromos, si es que hay activo algún juego de colección.

  - **Juego de cuestionario:** el juego de cuestionarios ha sido desarrollado paralelamente a nuestro proyecto por otro alumno.

  - **Juego de geocaching**: el juego de geocaching ha sido desarrollado paralelamente a nuestro proyecto por otro alumno

  - **Juego de avatar**: el juego de avatar ha sido desarrollado paralelamente a nuestro proyecto por otro alumno

4.1.2. Cuestionarios

La parte relativa a cuestionarios ha sido desarrollada por otro alumno en otro proyecto.

4.1.3. Colecciones

Permite al usuario acceder a las colecciones que ya tiene creadas o en su caso, crear nuevas.

- **Mis colecciones**: muestra las colecciones existentes y permite editarlas para modificar los parámetros que se definieron en el momento de la creación. Además, permite eliminar la colección seleccionada.

- **Crear colección**: permite al usuario definir la colección, elegir las imágenes de los cromos, determinar la probabilidad de aparición de cada cromo.

4.1.4. Puntos e Insignias

Permite al usuario acceder a los diferentes tipos de puntos que ya tiene creados o en su caso, crear nuevos.

- **Crear puntos e insignias**: cuando se crea un punto, se debe definir un nombre y una descripción. El valor del punto se le dará en el momento de asignarlo. Con las insignias sucede lo mismo, pero además será posible asignarle una imagen.

- **Mis puntos e insignias**: muestra los diferentes tipos de puntos y su descripción. Además, permite editar los parámetros que se definieron en el momento de la creación o eliminar el punto seleccionado. Lo mismo sucede con las insignias, además de mostrar la imagen si dispone de ella.

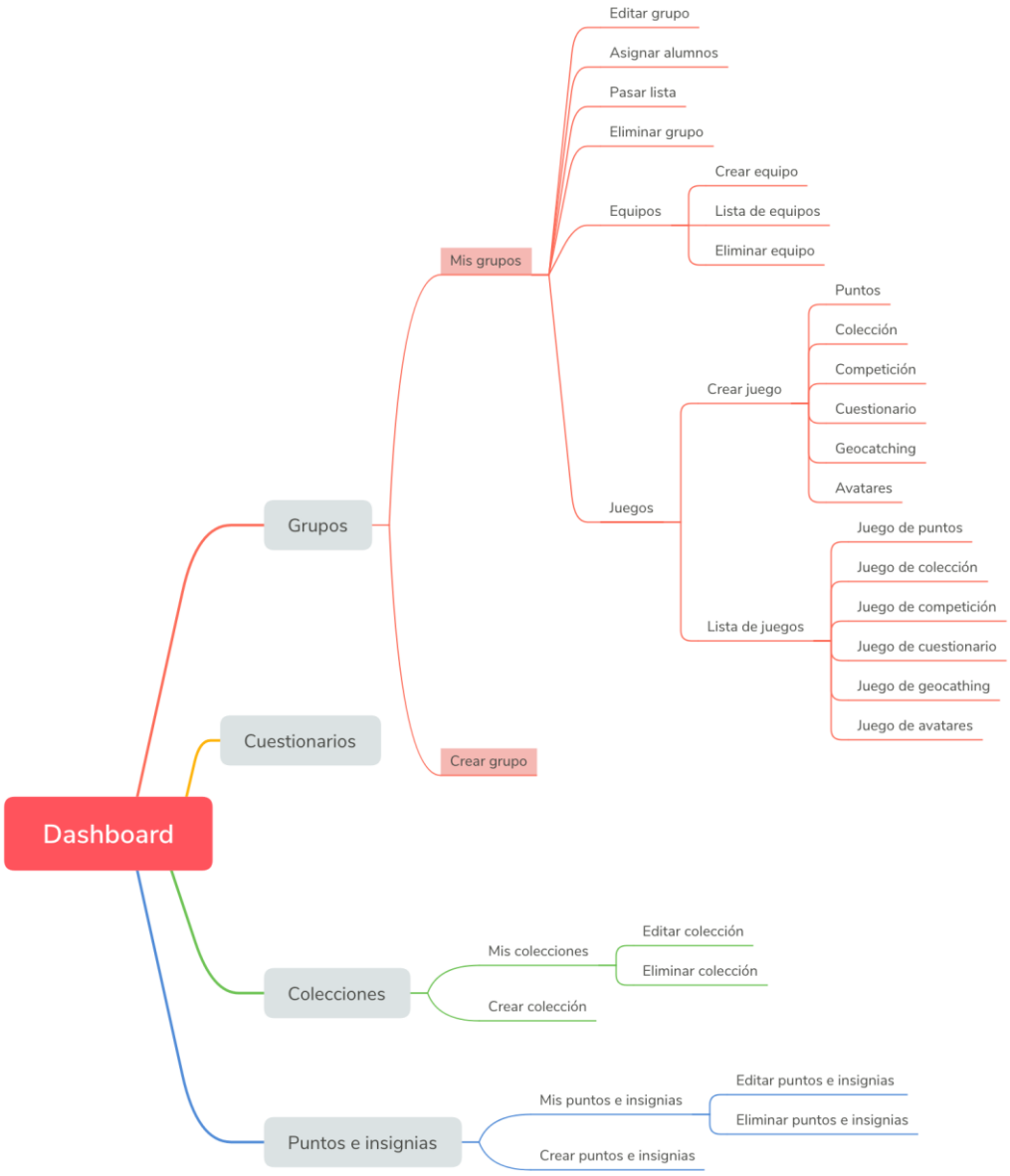


Figura 4.1 Mapa conceptual diseño funcional Dashboard

4.2. Mobile Profesor

Para visualizar el diseño funcional se hará uso de listas multinivel como se ha hecho con el Dashboard y se incluirá su respectivo mapa conceptual en la **Figura 4.2**. La aplicación se vuelve a organizar en los mismos cuatro bloques.

Es importante remarcar que el proyecto Mobile Profesor es un subconjunto de funcionalidades del Dashboard. Esto es debido a la ineficacia e incomodidad de realizar algunas acciones del Dashboard en el Mobile. Por ejemplo, es lógico y cómodo que un profesor pueda asignar puntos a un alumno desde el móvil, ya que se puede hacer de una manera fácil y rápida. Solo tenemos que seleccionar el tipo de punto, darle un valor y seleccionar el alumno.

En cambio, es ilógico incluir en el Mobile la opción de crear un grupo nuevo y añadirle todos los alumnos, ya que desde el móvil no podemos importar desde un archivo e introducirlos uno a uno manualmente sería largo y laborioso.

4.2.1. Grupos

Permite acceder a los grupos que fueron creados en el Dashboard.

Mis grupos: el usuario visualizará una lista con los grupos existentes y podrá seleccionar el que desee. Una vez seleccionado el grupo, aparecerá una pantalla que permitirá:

- **Miembros**: muestra los miembros que existen en ese momento en el grupo.

- **Equipos**: muestra los equipos que existen en ese momento, de manera que al seleccionar uno se abrirá un desplegable con el logo del equipo (si dispone de logo), los miembros del equipo.

- **Juegos**: mostrará una lista de los juegos activos e inactivos, permitiendo filtrar la lista por tipo de juego. Al elegir un juego podremos tomar todas las decisiones posibles en relación al juego.

    - **Juego de puntos**

      - Asignar cómodamente puntos a los alumnos/equipos (según el modo de juego), identificando el tipo de punto asignado.

      - Mostrar los rankings identificando los alumnos/equipos que están en cada nivel. Debería poder ver ranking total y rankings por tipos de puntos.

- **Juego de colección**

  - Asignar cómodamente cromos a los alumnos/equipos (según el modo de juego), identificando el cromo asignado o paquete de cromos.

  - Mostrar una lista de alumnos/equipos y al clicar poder identificar los cromos que tiene ordenados de mayor a menor nivel.

- **Juego de competición**

  - Asignar premios cómodamente de manera manual, identificando el tipo de punto asignado.

  - Mostrar clasificación de la liga en caso de jugar una competición tipo liga.

  - Mostrar el cuadro de clasificación en caso de jugar una competición tipo torneo de tenis.

  - Mostrar los resultados de los partidos, tanto si es modo liga como torneo de tenis.
  - Decidir los ganadores de los enfrentamientos de cada jornada. El profesor debería ir a la jornada en cuestión y encontrar las opciones para decidir, entre las cuales habría al menos:

    - A mano (debe ser cómodo señalar el ganador de cada emparejamiento).

    - Aleatoriamente.

    - El que tiene más puntos en el juego de puntos (si es que hay juego de puntos).

    - El que saco mejor puntuación en un juego de cuestionarios ya acabado (debería poder elegir el profe el juego de cuestionarios que debe usarse).

    - El que tenga más cromos de la colección de cromos, si es que hay activo algún juego de colección.

- **Juego de cuestionario:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.

- **Juego de geocaching:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.

- **Juego de avatar:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.

4.2.2. Cuestionarios

Como esta parte ha sido desarrollada por otro alumno en otro proyecto, no se ha incluido en el nuestro.

4.2.3. Colecciones

Permite al usuario acceder a las colecciones que ya tiene creadas o en su caso, crear nuevas.

- **Mis colecciones**: muestra las colecciones existentes.

4.2.4. Puntos e Insignias

Permite al usuario acceder a los diferentes tipos de puntos que ya tiene creados o en su caso, crear nuevos.

- **Mis puntos e insignias**: muestra los diferentes tipos de puntos y su descripción. Lo mismo sucede con las insignias, además de mostrar la imagen si dispone de ella.

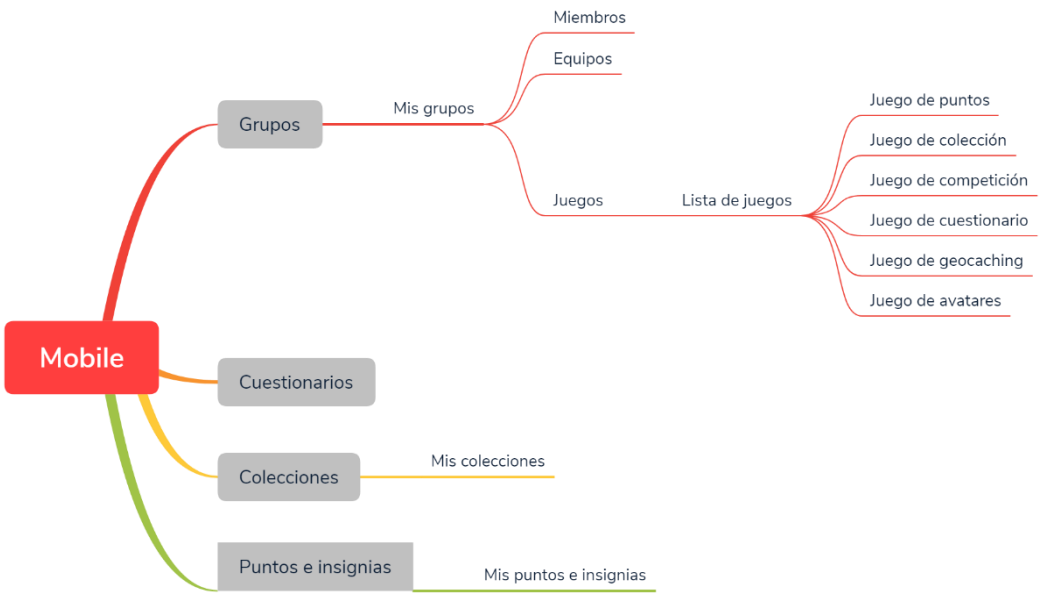


Figura 4.2 Mapa conceptual diseño funcional Mobile Profesor

4.3. Mobile Alumno

Esta aplicación se verá muy reducida con respecto a las dos anteriores debido a que se eliminarán muchas funcionalidades. Se dispone de un bloque a partir del cual se desarrolla la aplicación: Grupos. En la **Figura 4.3** se muestra en forma de mapa conceptual.

4.3.1. Grupos

Permite acceder a los grupos en los que el alumno está inscrito.

Mis grupos: el usuario visualizará una lista con los grupos existentes y podrá seleccionar el que desee. Una vez seleccionado el grupo, aparecerá una pantalla que permitirá:

- **Miembros**: muestra los miembros que existen en ese momento en el grupo.

- **Equipos**: muestra los equipos que existen en ese momento, de manera que al seleccionar uno se abrirá un desplegable con el logo del equipo (si dispone de logo), los miembros del equipo.

- **Juegos**: mostrará una lista de los juegos activos, permitiendo filtrar la lista por tipo de juego. Al elegir un juego podremos visualizar el progreso.

    - **Juego de puntos**

      - Mostrar los rankings identificando los alumnos/equipos que están en cada nivel. Pudiendo ver ranking total y rankings por tipos de puntos y ver los tipos de puntos y los requisitos para conseguir cada tipo de puntos, las condiciones para alcanzar cada nivel y los privilegios que se consiguen al alcanzar cada nivel.

    - **Juego de colección**

      - Mostrar cómo está su álbum de cromos (o el de su equipo). Además, podrá intercambiar cromos.

    - **Juego de competición**

      - Mostrar clasificación de la liga en caso de jugar una competición tipo liga.

      - Mostrar el cuadro de clasificación en caso de jugar una competición tipo torneo de tenis

- Mostrar los resultados de los partidos, tanto si es modo liga como torneo de tenis.

- También debería ver el calendario y enfrentamientos futuros, así como la descripción de los criterios que usará el profe para determinar los ganadores en cada jornada.

- **Juego de cuestionario:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.
- **Juego de geocaching:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.
- **Juego de avatar:** juego desarrollado paralelamente a nuestro proyecto por otro alumno.

Figura 4.3 Mapa conceptual diseño funcional Mobile Alumno

CAPÍTULO 5. DISEÑO PANTALLAS

Basándonos en el diseño funcional de los tres tipos de proyectos, construimos prototipos de pantallas respectivamente para cada uno de ellos. Esto lo hicimos para asegurarnos de que la información que mostramos en cada una de las pantallas se ajustaba a lo deseado antes de inicializarnos con el código. Un objetivo primordial era crear una aplicación intuitiva que guiara al usuario y que éste pudiera realizar las tareas necesarias haciendo los menos clics posibles.

Los prototipos tenían las funciones más básicas que deberían disponer las pantallas. Durante el desarrollo de la aplicación añadimos funcionalidades y componentes nuevos que mejoraban la aplicación tanto funcional como visualmente. No obstante, durante el proyecto se ha intentado seguir estos prototipos. Con lo que siempre podremos ver similitudes entre los prototipos y la realidad de las pantallas.

Es importante remarcar el hecho de que en el apartado de diseño funcional aparece el Mobile Alumno mientras que en este apartado no aparecerá. Esto se debe a que, como se explicó en el capítulo 3 (Objetivos y plan de trabajo), inicialmente se marcaron unos objetivos que incluían la realización del Mobile Alumno, pero estos cambiaron al tener que realizar todo el proyecto Classpip desde cero.

Como la decisión de empezarlo desde el principio se tomó semanas más tardes de empezar el proyecto, algunos apartados como el diseño funcional ya se habían realizado. En ese momento se decidió centrarse en los proyectos que se iban a realizar y dejar de lado el Mobile Alumno.

5.1. Dashboard

Para el diseño de las pantallas del Dashboard, al ser una aplicación web, se ha utilizado Wix, una plataforma que permite el desarrollo de páginas web online de manera muy sencilla.

Pese a que el Dashboard no es una página web, para crear una maqueta de cómo va a ser nuestra aplicación, Wix nos sirve a la perfección y es mucho más sencillo de usar que otras opciones disponibles para el diseño de aplicaciones web.

No se han llevado a cabo el maquetado de todas las pantallas ya que hemos seguido siempre un mismo formato a lo largo de todo el diseño. Al final estos diseños, validaban el diseño funcional y se mostraban las funcionalidades en cada pantalla de una manera clara. A medida que se ha ido avanzando en el proyecto se han proyectado nuevas ideas para mejorar la aplicación que no mostramos en los prototipos.

Como se puede apreciar en la figura del mapa conceptual del diseño del Dashboard, el elemento que contiene más ramas y complejidad es el apartado de “Grupos”, concretamente en “Mis Grupos”. Por ello, nos centraremos básicamente en este apartado en el prototipo.

Inicialmente se diseñó el *navbar*, el cual debía contener accesos rápidos a los componentes principales. Así pues, este incluye un botón de “Classpip”, otro de “Inicio” y desplegables de los 4 bloques principales que contienen dos subfunciones: crear y la listar. En la **Figura 5.1** podemos ver el prototipo de la barra de navegación.

Figura 5.1 Prototipo navbar

El prototipo de las pantallas “Classpip” e “Inicio” no se ha llevado a cabo, consecuentemente nos disponemos a comentar el primer bloque, el de “Grupos”.

5.1.1. Prototipos pantallas “Grupos”

En el desplegable de “Grupos” encontraremos tanto “Crear Grupo” cómo “Mis Grupos”.

Crear grupo: asigna un nombre al grupo e indica su curso y asignatura (descripción). Con estos dos parámetros ya se podrá generar el grupo, sin embargo, también se permite la opción de agregar alumno al grupo. La **Figura 5.2** muestra el prototipo de pantalla de esta función.

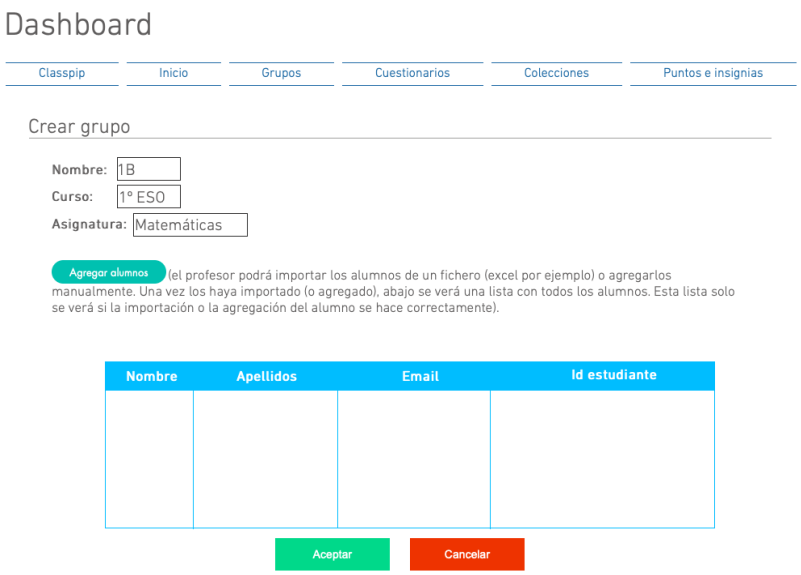


Figura 5.2 Prototipo pantalla “Crear Grupo”

Mis Grupos: muestra una lista con los grupos anteriormente creados y permite acceder a un grupo cuando se haga *clic* encima. En la **Figura 5.3** se muestra el listado de grupos de un profesor.

Figura 5.3 Prototipo pantalla “Mis Grupos”

Grupo Seleccionado: al seleccionar un grupo, se visualiza la información del mismo y todas las funciones que se pueden efectuar en él como aparece en la **Figura 5.4**. Para acceder rápidamente a estas funciones, se dispone de botones situados en la parte superior de la lista indicando claramente que realiza cada uno de ellos: editar grupo, pasar lista, equipos, juegos y eliminar grupo.

Figura 5.4 Prototipo pantalla "Grupo Seleccionado"

Editar grupo: permite modificar cualquier parámetro del grupo al igual que cuando se creó. Esto permite modificar su nombre, la descripción y añadir alumnos como aparece en la **Figura 5.5**. Además, también se podrá eliminar alumnos del grupo. Nuevamente, estas dos últimas funciones serán accesibles a partir de botones.

Figura 5.5 Prototipo pantalla "Editar Grupo"

Equipos: para no realizar un botón desplegable para diferenciar entre la lista de equipos y crear equipo, se utilizará algún método que visualizase una opción u otra en función de un clic. A la hora de realizar el prototipo, muestra este funcionamiento con dos botones: uno para visualizar la lista (**Figura 5.6**) y otro para crear un nuevo equipo (**Figura 5.7**).

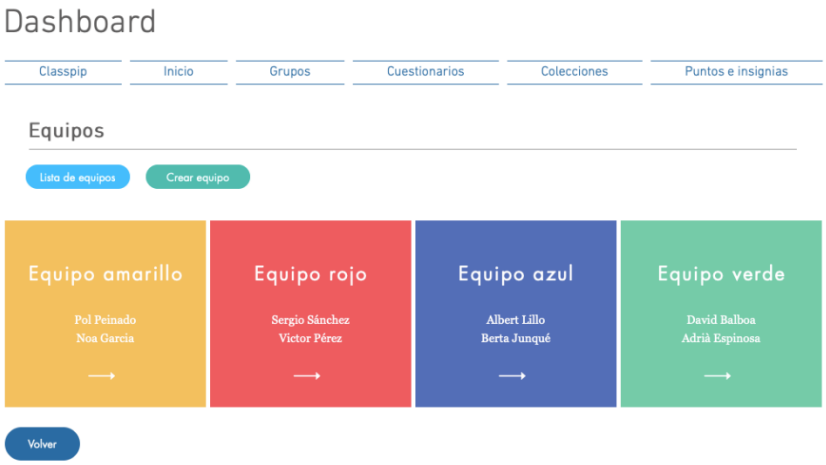


Figura 5.6 Prototipo pantalla "Equipos"

La pantalla de crear equipo se diseñó posteriormente a la de crear grupo. En un principio se quería mantener la misma estructura en ambas pantallas, pero durante el diseño se estudiaron los componentes de Angular Material que podríamos utilizar para realizar estas pantallas. Se encontró uno que era muy útil para procesos de varios pasos, como puede ser la creación de algo. Este componente se llama *stepper*.

En el prototipo de Wix se decidió representar una especie de *stepper* con dos botones, indicando cada uno de los pasos. Consideramos dejar estas pantallas sin cambiar para mostrar la evolución en el proceso de diseño.

Figura 5.7 Prototipo pantalla "Crear Equipo"

Equipo seleccionado: se accede a un equipo haciendo *clic* en él, al igual que sucedía cuando se diseña la lista de mis grupos. Esta pantalla mostrada en la **Figura 5.8**, nuevamente, mostrará las funciones que podrá hacer con el equipo: editar y eliminar.

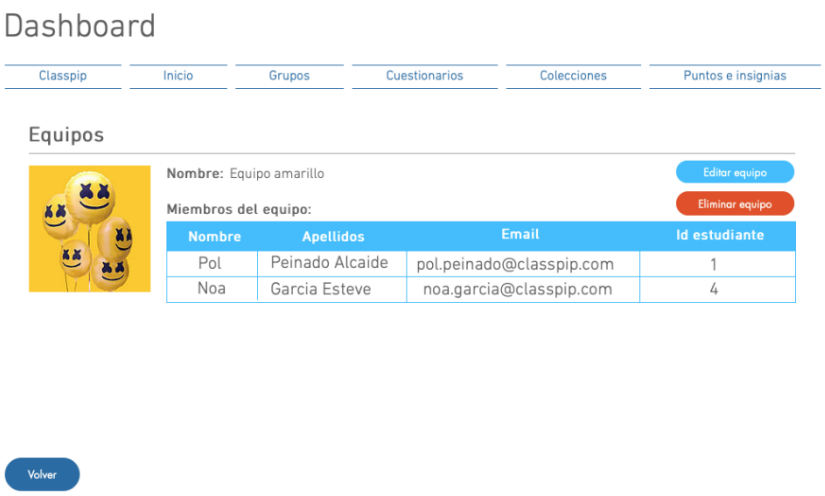


Figura 5.8 Prototipo pantalla "Equipo Seleccionado"

Editar equipo: realiza las modificaciones referentes a los parámetros generados (nombre y alumnos). Además de borrar o agregar alumnos, se dispone de un botón que permitirá mover alumnos de un equipo a otro del grupo de manera sencilla. El prototipo se muestra en la **Figura 5.9**.

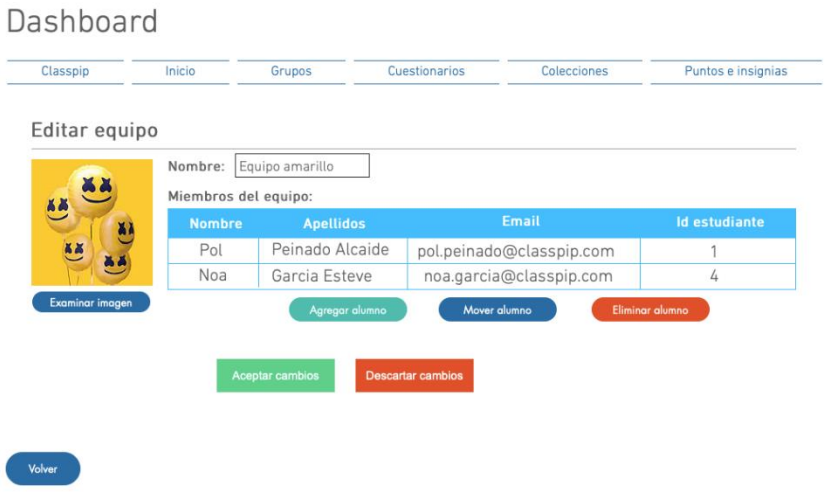


Figura 5.9 Prototipo pantalla "Editar Equipo"

Juegos: se debe retroceder para poder acceder al apartado “Juegos” del grupo seleccionado. Al igual que en equipos, se utiliza un mismo componente para mostrar la lista de juegos (**Figura 5.10**) y crear uno nuevo (**Figura 5.11**).

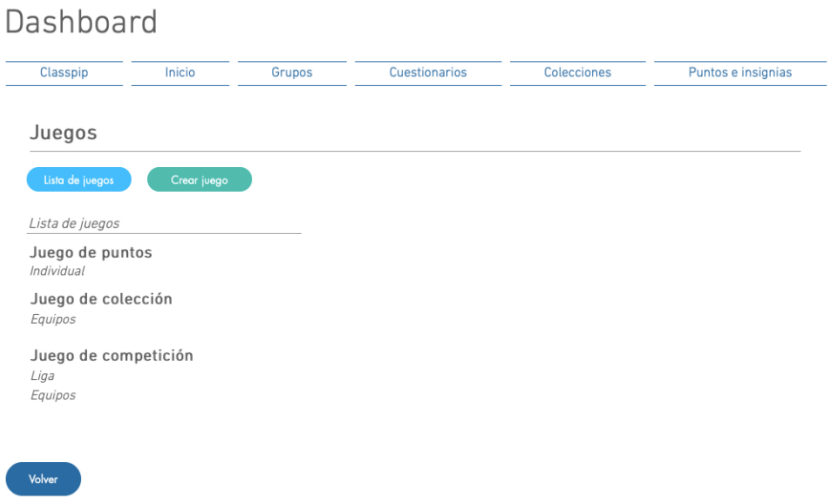


Figura 5.10 Prototipo pantalla "Juegos"

Al igual que cuando se genera un equipo, el proceso de creación de un juego constará de varios pasos. En el primero, se deberá escoger el tipo de juego con el que se desea jugar, y en función del escogido saldrá una opción u otra.

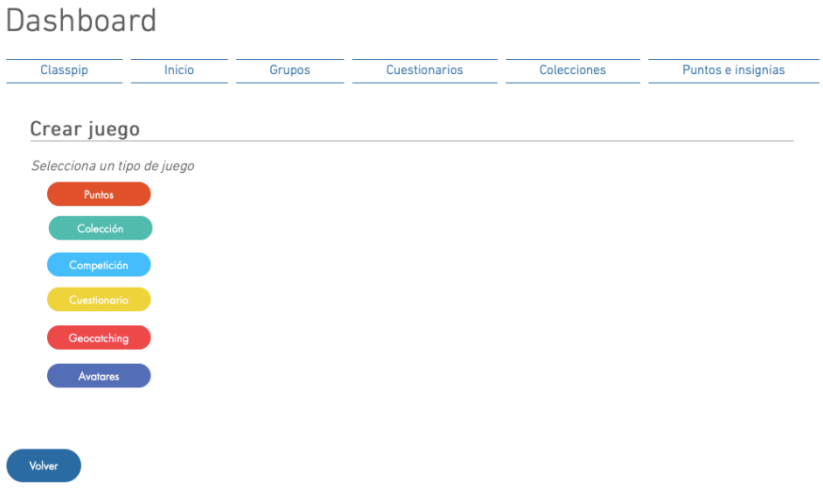


Figura 5.11 Prototipo pantalla "Crear juego"

Hasta el momento solo hay implementados tres tipos de juegos: de puntos, colección y competición. Por consiguiente, solo se realizará el diseño de estas tres pantallas.

Juego de puntos: permite añadir los puntos al juego y eliminarlos si no son deseados como muestra la **Figura 5.12**. Además, el profesor podrá monitorizar los puntos que esté añadiendo al juego. También se debe escoger si se desea generar un juego individual o por equipos.

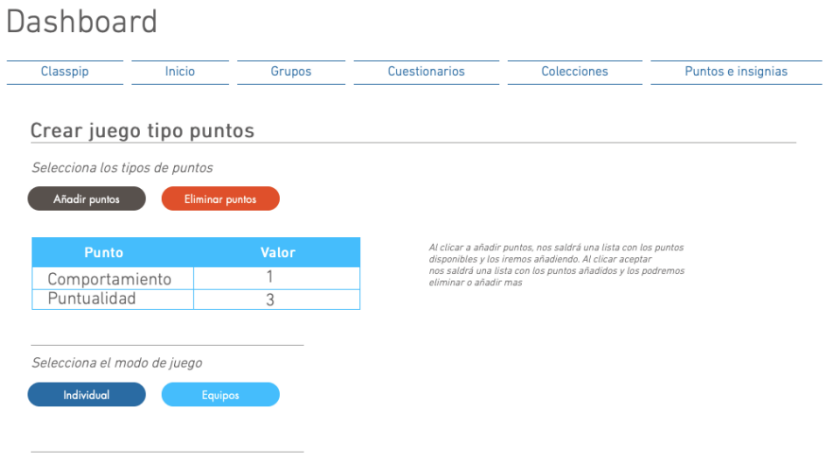


Figura 5.12 Prototipo pantalla "Crear juego" de puntos

Juego de colección: permite seleccionar la colección con la que deseamos jugar (**Figura 5.13)**. Esto se propone hacer a través de un desplegable que muestre las colecciones. También se escoge el modo de juego a través de botones. Finalmente se seleccionan reglas automatizadas de asignación de cromos. Se irán listando las reglas seleccionadas para el juego.

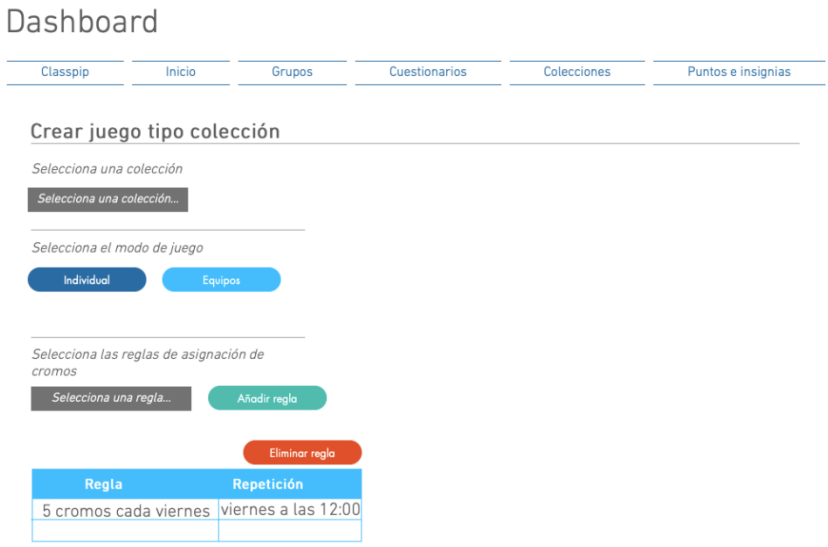


Figura 5.13 Prototipo pantalla "Crear juego" de colección

Juego de competición: permite seleccionar el tipo de competición y el modo del juego a través de botones. Después se eligen los criterios de ganador de la jornada o del partido. Por último, se escogen las reglas automatizadas del juego a partir de un desplegable. El prototipo de la creación de este juego se puede ver en la **Figura 5.14**.

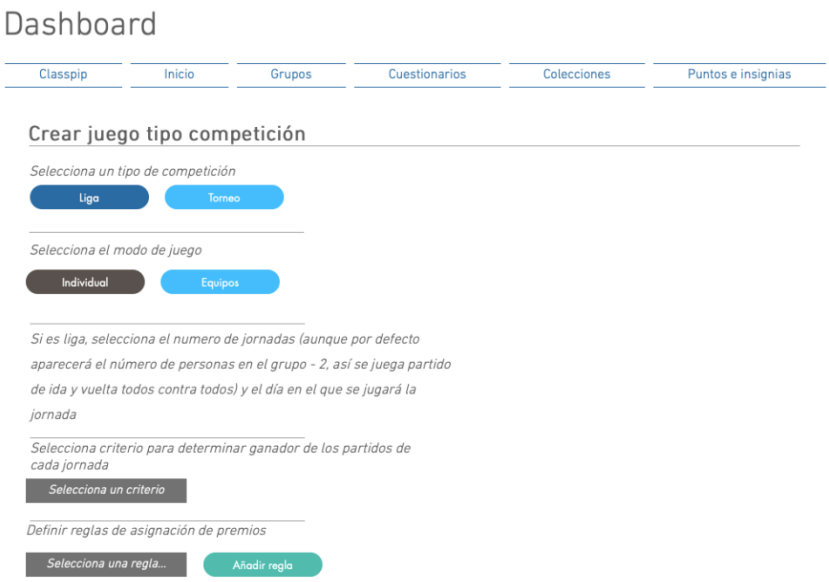


Figura 5.14 Prototipo pantalla "Crear juego" de competición

5.2. Mobile Profesor

Para el diseño de las pantallas del Mobile, al ser una aplicación móvil, se ha utilizado MockingBot, una plataforma que permite el desarrollo de aplicaciones móvil online de manera sencilla e intuitiva.

No se han llevado a cabo el maquetado de todas las pantallas ya que hemos seguido siempre una misma idea a lo largo de todo el diseño, con lo que se ha evitado diseñar pantallas iguales o muy parecidas. A partir de este momento

Tanto en Dashboard como en Mobile, se puede apreciar en la figura del mapa conceptual de ambas, lo que tiene más ramas y complejidad es el apartado de “Grupos”, concretamente en “Mis Grupos”. Por eso nos centraremos básicamente en este apartado en el prototipo.

5.2.1. Prototipos pantallas “Grupos”

Mis Grupos: muestra una lista con los grupos creados por el profesor y permite acceder a un grupo cuando se hace *clic* encima. El prototipo se puede ver en la **Figura 5.15**.

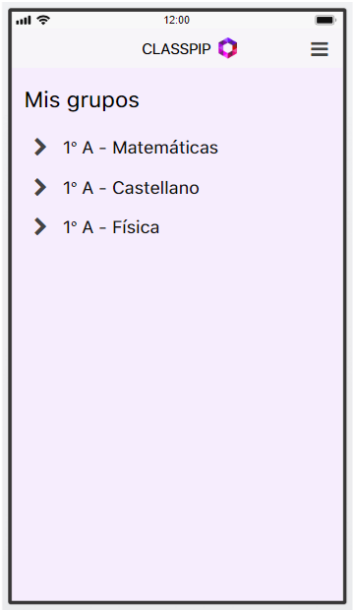


Figura 5.15 Prototipo pantalla lista "Mis grupos"

Grupo seleccionado: al hacer *clic* encima del grupo, se abre un menú en acordeón donde se muestran todas las funciones del grupo. Para acceder rápidamente a estas funciones solo hace falta hacer *clic* encima de las mismas. El prototipo se puede ver en la **Figura 5.16**.

Figura 5.16 Prototipo pantalla grupo seleccionado dentro de "Mis grupos"

Alumnos: al hacer *clic* encima de “Alumnos”, se abre una nueva página en la que se muestran todos los alumnos del grupo. El prototipo se puede ver en la **Figura 5.17**.

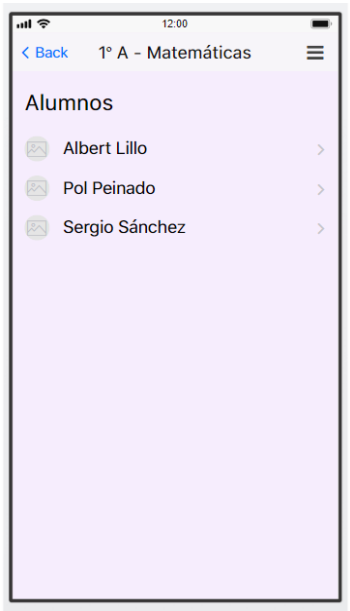


Figura 5.17 Prototipo pantalla "Lista de alumnos"

Equipos: al hacer *clic* encima de “Equipos”, se abre una nueva página en la que se muestran todos los equipos del grupo y su logo. El prototipo se puede ver en la **Figura 5.18**.

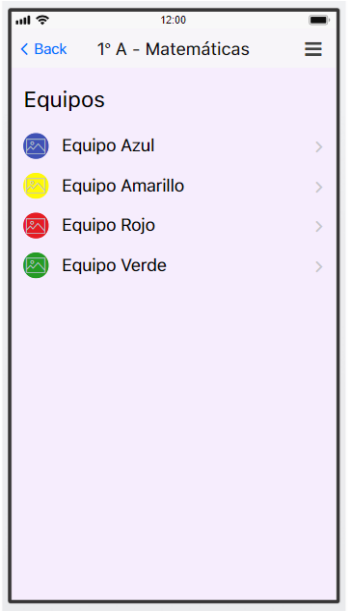


Figura 5.18 Prototipo pantalla "Lista de equipos"

Equipo seleccionado: al seleccionar un equipo se mostrarán los alumnos que pertenecen al equipo. El prototipo se puede ver en la **Figura 5.19**.

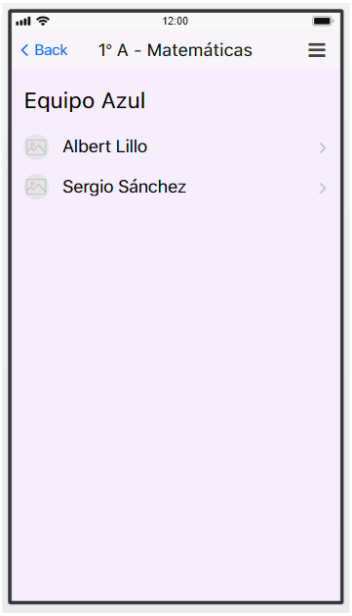


Figura 5.19 Prototipo pantalla equipo seleccionado

Juegos: al hacer *clic* encima de “Juegos”, se abre un menú en acordeón donde se muestran todos los juegos activos del grupo. En el ejemplo que se muestra en la Figura 5.19. el juego de “Clasificación por Puntos” representa al juego de puntos, el de “Cromos de Matemáticos” representa al juego de cromos y el juego de “Liga de la Justicia” representa al juego de competiciones. El prototipo se puede ver en la **Figura 5.20**.

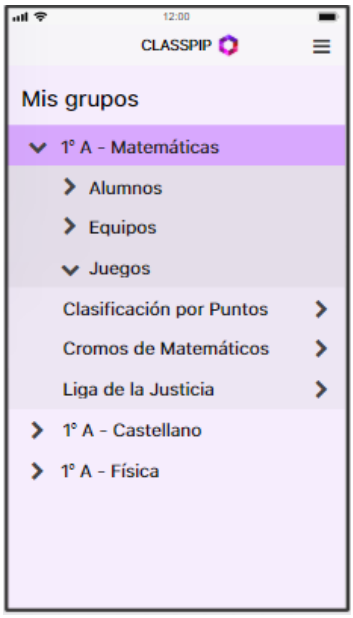


Figura 5.20 Prototipo pantalla "Lista de juegos"

Hasta el momento solo hay implementados tres tipos de juegos: de puntos, colección y competición. Por consiguiente, solo se realizará el diseño de estas tres pantallas.

Juego de puntos: al seleccionar el juego de puntos, se muestra la clasificación (en este ejemplo se muestra un juego de puntos individual) por puntos totales, pudiendo elegir la clasificación por tipo de punto específico. En la parte inferior se encuentra la opción de asignar punto a un alumno o grupo, dependiendo del modo de juego. El prototipo se puede ver en la **Figura 5.21**.

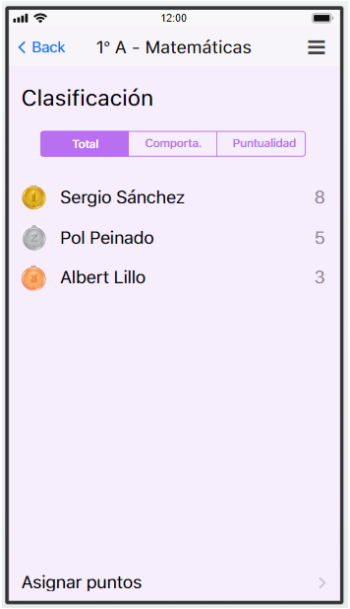


Figura 5.21 Prototipo pantalla "Juego de Puntos"

Juego de colección: permite asignar un paquete de cromos aleatorios a miembro del grupo que el profesor quiera. Además, se puede elegir el número de cromos por paquete. El prototipo se puede ver en la **Figura 5.22**.

Figura 5.22 Prototipo pantalla "Juego de Colección"

Juego de competición: permite visualizar la clasificación, el calendario y elegir a los ganadores de la jornada. En el apartado de clasificación se muestran todos los miembros del grupo ordenados por victorias, además podemos dirigirnos rápidamente al calendario desde un botón, en el calendario se muestra una jornada con los partidos y el criterio que se ha tenido en cuenta para seleccionar el ganador. El prototipo se puede ver en la **Figura 5.23**.

Figura 5.23 Prototipo pantalla "Juego de competición"

CAPÍTULO 6. DISEÑO BASE DE DATOS

El hecho de considerar que la aplicación será una herramienta diseñada para el profesor (no para la escuela) y que algunas de las funcionalidades que se han añadido son nuevas, hacía que la base de datos cambiara significativamente, en especial los modelos y las relaciones entre ellos. Como era más complejo modificar todas las propiedades de los modelos, revisar que los modelos que había eran los que necesitábamos, añadir los que faltaban y revisar las relaciones entre ellos, decidimos empezar un nuevo proyecto Services.

Para diseñar correctamente la API, inicialmente se han examinado los prototipos de pantallas del Dashboard y las consultas que se llevarían a cabo. Se hizo con este proyecto porque el Mobile Profesor no es más que una versión reducida de las funcionalidades del Dashboard. Lo que cambiará significativamente entre ambos proyectos es que, para la interfaz gráfica, uno usa Angular Material mientras que el otro usa Ionic.

Como se explicó en el apartado de “Diseño de pantallas”, éstas no son más que maquetas muy simples con sus funciones básicas. A medida que se han ido montando las pantallas en el Dashboard y hemos tenido nuevas ideas para implementar, hemos añadido nuevos modelos y relaciones que a priori no contemplábamos.

Es importante remarcar que, al crear un nuevo proyecto Services, se creó una base de datos en memoria llamada *baseDatosEnMemoria*, donde se crean todos los modelos y se almacenan los datos. Estos datos pueden visualizarse en el archivo ***db.json*** de la carpeta ***data***.

6.1. Modelos

El concepto de modelo es importante para entender mejor el diseño de la API. Los modelos son una representación de los datos. Para entenderlo mejor, son una especie de clases de la API REST. Los modelos nos permiten guardar la información en sus respectivas variables. Podremos encontrar los modelos creados en el proyecto Services en la carpeta ***models*** dentro de ***common***. Los archivos importantes son los que tienen la extensión *json*. En ellos podremos ver las propiedades y las relaciones de los modelos.

A lo largo del trabajo se han creado un total de 32, como por ejemplo el modelo Profesor, Alumno, Juego, JuegoDePuntos, AlumnoJuegoDePuntos, etc. Algunos de los modelos creados no se han utilizado, ya que no ha dado tiempo a implementar las funcionalidades para los que fueron diseñados.

Como hay demasiados modelos, escogeremos los modelos citados anteriormente, ya que luego nos servirán para explicar las dos relaciones más

importantes que hay entre modelos. Los demás modelos creados se recogerán en el anexo.

6.1.1. Modelo Profesor

Los pasos a seguir en la creación de nuevos modelos son muy sencillos y los podemos encontrar en la Lección 4: Servicios.

Cuando creamos un nuevo modelo en la base de datos que hemos creado en memoria, el siguiente paso es escoger la clase base del modelo. Esto significa coger un modelo y sus atributos como base del modelo que estamos creando. Lo más común será seleccionar la opción PersistedModel, lo cual te permite crear un modelo limpio. No obstante, habrá situaciones en las que deberemos escoger como base otro modelo.

Para que quede más claro el concepto de base, pondremos un ejemplo con el modelo Profesor. El modelo profesor actualmente está utilizando una base PersistedModel para evitar errores de permisos. Sin embargo, en un futuro deberá utilizar como base el modelo User que incluye la API REST por defecto.

Este modelo User dispone de las propiedades mostradas en la tabla inferior.

Tabla 6.1 Propiedades modelo User

User
“realm”
“username”
“email”
“emailVerified”
“id”[1]

A continuación, nos disponemos a crear las propiedades del modelo Profesor. Cuando el profesor se registre en la aplicación deberá identificarse y podrá cambiar la foto de perfil que se le asigna por defecto para personalizar su aplicación. Las propiedades del modelo profesor se recogen a continuación:

Tabla 6.2 Propiedades modelo Profesor

Profesor
“Nombre”
“Apellido”
“ImagenPerfil”

[1] La propiedad id viene por defecto en cada modelo, por lo que no se añadirán en los siguientes modelos

Una vez creado el modelo, podremos observar que él recoge las propiedades tanto del modelo User como del propio modelo profesor. Esto nos será útil a la hora de crear juegos, ya que todos utilizarán una misma base.

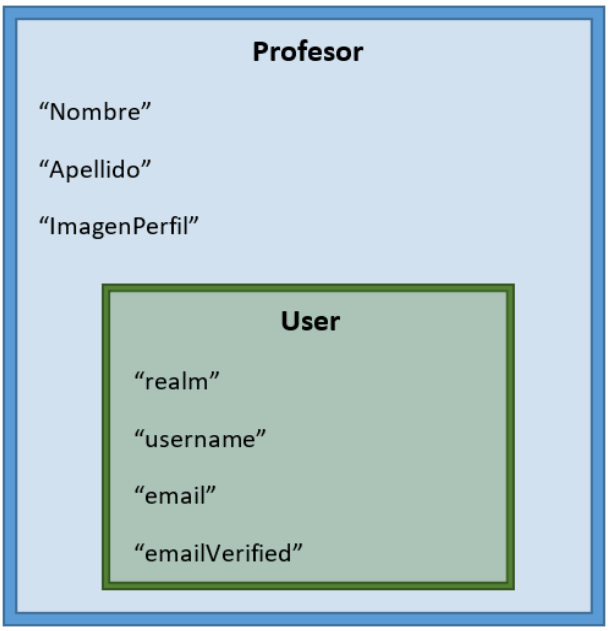


Figura 6.1 Modelo Profesor con base User

Como se ha explicado previamente, a lo largo de este proyecto se ha utilizado la base PersistedModel tanto para el modelo Profesor como para el Alumno, ya que no ha dado tiempo a implementar la **autentificación autentica**.

6.1.2. Modelo Alumno

El modelo Alumno será muy similar al modelo Profesor. En un futuro también, deberá utilizar la base User, pero por el momento hace uso de PersistedModel. En este caso es importante incluir tanto el primer como el segundo apellido para reducir la posibilidad de que el nombre de dos alumnos coincida.

Tabla 6.3 Propiedades modelo Alumno

Alumno
“Nombre”
“PrimerApellido”
“SegundoApellido”
“ImagenPerfil”

6.1.3. Modelo Juego

El modelo Juego se genera con la funcionalidad de recopilar las propiedades en común entre los diferentes tipos de juego. Esto se hace ya que se utilizará este modelo como base para los modelos de los diferentes tipos de juegos. Es decir, los modelos JuegoDePuntos, JuegoDeColeccion y JuegoDeCompeticion serán de la base Juego.

Las propiedades en común que tienen todos los juegos son el tipo de juego (nos permite identificar el juego al que nos referimos), el modo (individual o por equipos) y el estado (activo o inactivo).

Tabla 6.4 Propiedades modelo Juego

Juego
“Tipo”
“Modo”
“JuegoActivo”

6.1.4. Modelo JuegoDePuntos

En este caso, el modelo JuegoDePuntos no necesitará de ninguna propiedad adicional a parte de las propiedades de la base Juego.

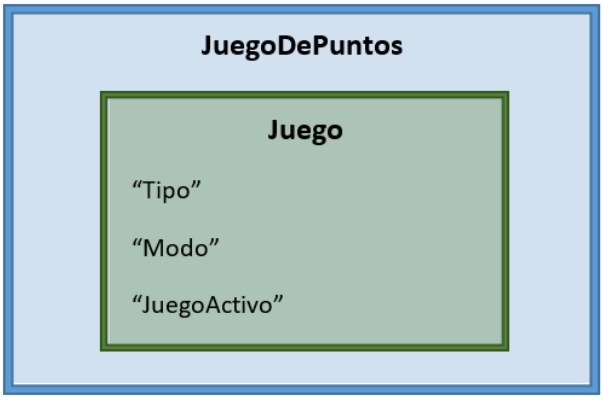


Figura 6.2. Modelo JuegoDePuntos con base Juego

6.1.5. Modelo AlumnoJuegoDePuntos

Este modelo se utiliza como puente entre el modelo Alumno y el modelo JuegoDePuntos en una relación N:M. El modelo relaciona un alumno en concreto con un juego de puntos en concreto. Además, incluirá el valor de los puntos que lleva el alumno en ese juego de puntos.

Tabla 6.5 Propiedades modelo AlumnoJuegoDePuntos

AlumnoJuegoDePuntos
"PuntosTotalesAlumno"

A continuación, en el apartado relaciones, se expondrán los diferentes tipos de relaciones disponibles y se explicará más en detalle para que utilizaremos este modelo.

6.2. Relaciones entre modelos

Los modelos por si solos solo nos permiten almacenar información en su variable correspondiente. En una aplicación real, los modelos suelen estar conectados o relaciones con otros modelos. Es por eso que se necesita definir relaciones entre ellos.

Con los modelos conectados, LoopBack permite consultar y filtrar la información según las necesidades del usuario. Se pueden definir las siguientes relaciones entre modelos:

- **BelongsTo:** relación de varios modelos a uno (N:1). El modelo declarador puede tener como máximo una instancia del otro modelo, mientras que el modelo destino puede tener muchos del modelo declarador.

- **HasOne:** relación uno a uno con otro modelo (1:1), de forma que cada instancia del modelo declarador tiene una instancia del otro modelo.

- **HasMany:** relación de uno a varios modelos (1:N). Cada instancia del modelo tiene cero o más instancias de otro modelo.

- **HasAndBelongsToMany:** relación de varios a varios (N:M), sin ningún tipo de modelo intermedio.

Como cada modelo puede disponer de varias relaciones, explicarlas todas resulta imposible. Las recogeremos todas en el **Anexo 12.2** junto a las

propiedades de los modelos. Siguiendo con los modelos anteriores, explicaremos las relaciones utilizadas.

6.2.1. Relación Profesor – Alumno

A la hora de decidir qué tipo de relación se debe utilizar, se deben realizar dos afirmaciones:

1. Un profesor[2] tiene varios alumnos[3].

2. Un alumno tiene varios profesores.

Si la respuesta a una afirmación es correcta significa que la relación es 1:N. En el caso de que la respuesta sea correcta en ambos casos, significa que la relación es N:M.

Como se ha remarcado a lo largo de la memoria, la herramienta Classpip es una herramienta enfocada a un profesor, no para una escuela. Consecuentemente, cada alumno tendrá un profesor solo.

Por consiguiente, un profesor tendrá varios alumnos, pero un alumno solo tendrá un profesor. Con lo que la relación que utilizaremos es la 1:N.

Al crear esta relación podremos especificar una clave foránea. Los modelos se relacionan con sus identificadores únicos, con lo que la clave foránea siempre serán los IDs correspondientes. En este caso, la clave foránea será el profesorId.

La clave foránea se podrá visualizar entre las propiedades del modelo alumno cuando la relación este hecha. Así pues, cuando creemos un alumno, además de indicar su nombre y apellidos, deberemos especificar de qué profesor es alumno.

Por ejemplo, imaginemos la situación que registramos al profesor Miguel Valero en Classpip, el cual recibe el identificador 1 (profesorId). Miguel decide incluir a tres alumnos a los cuales imparte clase, Pol Peinado, Sergio Sánchez y Albert Lillo. Al crearlos, deberá especificar que estos alumnos pertenecen al profesor cuyo identificador es el 1.

Más adelante se muestra (**Figura 6.4**) un ejemplo de las consultas a la base de datos necesarias en este ejemplo.

[2] El nombre del primer modelo en singular
[3] El nombre del segundo modelo en plural

6.2.2. Relación Alumno – JuegoDePuntos

Se vuelven a plantear las mismas afirmaciones anteriores, pero con sus respectivos modelos.

1. Un alumno puede jugar varios juegos de puntos.

2. Un juego de puntos puede ser jugado por varios alumnos.

En este caso, ambas afirmaciones son correctas, por lo que la relación es N:M. Al principio intentamos crear la relación N:M mediante el uso del HasAndBelongsToMany. No obstante, no hubo éxito al comprender cómo funcionaba, así que se hizo uso de otro método para generar este tipo de relaciones. Este método involucra cuatro relaciones y un modelo intermedio. Este modelo intermedio es el modelo llamado AlumnoJuegoDePuntos.

Al disponer de varios juegos de puntos y varios alumnos, se debe relacionar un alumno determinado con un juego de puntos en concreto. Con esto podremos ver el progreso de ese alumno en ese juego. Esa relación se hace en el modelo intermedio que hemos creado. En la siguiente figura se representan estas relaciones.

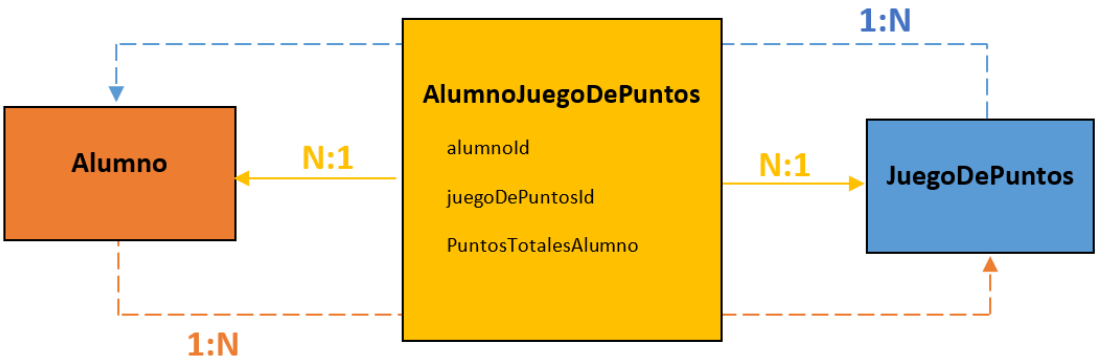


Figura 6.3 Representación gráfica relación N:M

Primero se realiza una relación 1:N (hasMany) entre el modelo Alumno y JuegoDePuntos through AlumnoJuegoDePuntos (línea naranja). Posteriormente se realiza lo mismo, pero al revés (línea azul). Finalmente haremos dos relaciones belongsTo de AlumnoJuegoDePuntos a Alumno y JuegoDePuntos respectivamente (líneas amarillas). Este proceso lo podemos encontrar en los videos de la Lección 18: Relaciones N:M.

Pongamos el siguiente ejemplo: imaginemos que se crea un juego de puntos cuyo identificador es el 1 (juegoDePuntosId). Los alumnos del ejemplo anterior habían recibido los identificadores 1, 2 y 3 (alumnoId) respectivamente. Con lo que, cuando se inscribió a los alumnos en el juego de puntos, se vincularon a

los tres alumnos en tres modelos diferentes de AlumnoJuegoDePuntos modificando el alumnoId, mientras que el identificador del juego sigue siendo el mismo.

6.3. Consultas a la base de datos

Las relaciones hechas en el apartado anterior nos permiten hacer las consultas adecuadas a la base de datos. Debemos tener claro al menos cinco métodos de petición HTTP:

- **GET:** nos devuelve la información.
- **POST:** nos permite crear una nueva información (una nueva entrada).
- **PUT:** permite modificar valores de una entrada ya creada. No obstante, deberemos introducir de nuevo todos los valores de los parámetros.
- **PATCH:** permite modificar valores concretos de una entrada introduciendo solo el nuevo valor.
- **DELETE:** nos permite borrar una entrada almacenada en la base de datos.

Pondremos dos ejemplos de consultas que conseguimos con las dos relaciones explicadas anteriormente.

6.3.1. Consulta añadir alumno al profesor

El profesor debió incluir a los tres alumnos a los que daba clase. Para ello lo único que se debe hacer son tres POSTs rellenando las propiedades del modelo alumno correspondientemente. Hay dos métodos para realizar el POST según la URL:

- Mediante la URL de Alumnos e introducir el profesorId junto con las propiedades.
- Mediante la URL de Profesores, lo cual deberemos indicar el identificador del profesor en la misma URL. Por consiguiente, no será necesario incluir el profesorId con sus propiedades.

Para nuestro caso, se ha hecho uso del segundo método. A continuación, se muestra un ejemplo del primer de los tres POSTs necesarios.

Figura 6.4 Ejemplo POST alumnos de un profesor

6.3.2. Consulta añadir alumno al juego de puntos

Para poder vincular un alumno a un juego de puntos, previamente se deberá realizar un POST del propio juego. Imaginemos que esto ya se ha realizado y que se ha generado el juego cuyo identificador es el 1.

Si se quisiera añadir los tres alumnos creados en el ejemplo anterior (identificadores 1, 2 y 3 respectivamente), lo único que se debería hacer son tres POSTs en el modelo AlumnoJuegoDePuntos. Mostramos el ejemplo de incluir a Pol en el juego de puntos 1. Para realizar el POST solo se deberá indicar el alumnoId y el juegoDePuntosId. Los puntos totales del alumno adquieren el valor por defecto 0.

Figura 6.5 Ejemplo inscripción alumno en juego de puntos

CAPÍTULO 7. ESTILO GRÁFICO

Se define como estilo gráfico toda la estructura gráfica que permite visualizar los elementos en la aplicación, es decir, la forma en que se estructura el aspecto de la aplicación. Debido al uso de distintas herramientas tanto en Dashboard como en Mobile, se han realizado los estilos gráficos de distinta forma como se puede observar a continuación.

7.1. Dashboard

Uno de los objetivos más importantes de este proyecto era centralizar todo el estilo gráfico en una misma página de estilos. Además, se ha tenido en cuenta la coherencia entre los estilos de los módulos, es decir, para todas las secciones de creación se utiliza un *stepper* y para las secciones donde se listan dichas creaciones se utilizan *Expansion Panel*. Además, se ha mantenido la coherencia entre los tipos de letra, tablas y botones.

La herramienta utilizada para organizar todos los elementos del estilo gráfico *Flex Layout*, esta herramienta permite organizar de manera sencilla e intuitiva los contenedores utilizados en la aplicación Dashboard. En la **Figura 7.1**, se observa un ejemplo de cómo se han utilizado los contenedores en la ventana de Mis Grupos.

Cada ventana tiene sus contenedores en el componente *.scss*, estos contenedores no se han centralizado en una misma página de estilo ya que están particularizados para cada ventana.

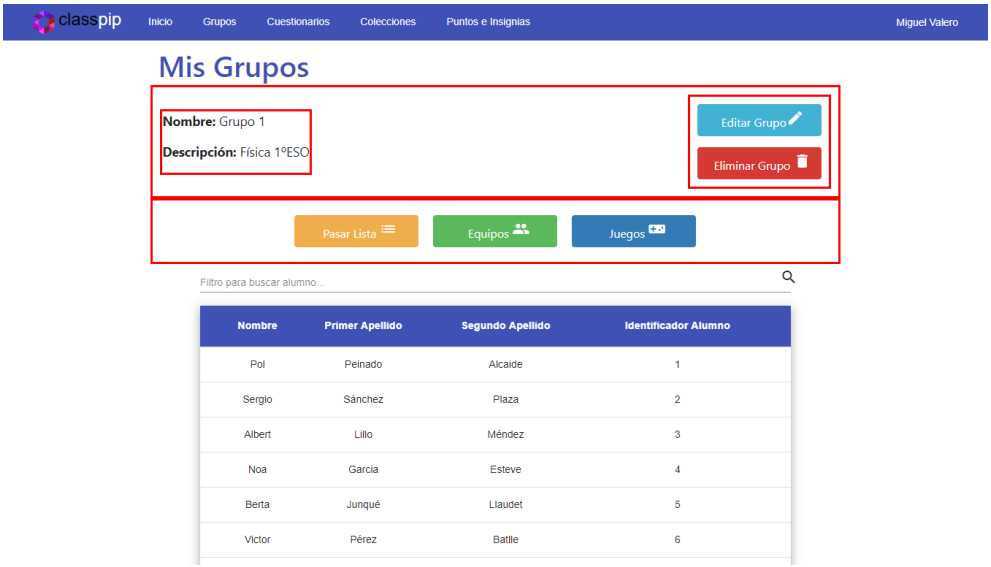


Figura 7.1 Ejemplo contenedores en pantalla “Mis Grupos”

En el componente llamado "*styles.scss"* se centralizan todos los estilos, como se puede ver a continuación incluye las opciones gráficas para títulos, tablas, botones, etc.

7.1.1. Títulos

Las opciones graficas de los títulos son las siguientes. Las clases "titulo" y "subtitulo" son utilizadas para posicionar y dar color del texto del título y subtítulo. La clase "h2" edita el tamaño de letra aumentándola y por último la clase "lineaDivisoria" posiciona la línea de debajo del título o subtítulo.

```
// Opciones gráficas título
.titulo{
  width: 70%;
  margin-left: 15%;
  height: 60px;
  padding: 10px;
  color: rgb(63, 81, 181);

}
.subtitulo{
  width: 70%;
  margin-left: 15%;
  height: 45px;
  padding: 10px;
  color: rgb(63, 81, 181);

}

.h2, h2 {
  font-size: 2.5rem !important;
}
.lineaDivisoria{
  width: 70%;
  margin-left : 15% !important;
}
```

Figura 7.2 Opciones gráficas títulos

En el componente *.html* se implementa el código de la **Figura 7.3** con el fin de conseguir el título deseado.

```
<div class="titulo"><h2>Mis Grupos</h2></div>
<mat-divider class="lineaDivisoria"></mat-divider>
```

Figura 7.3 Implementación en html de los títulos

Finalmente, el título obtiene la forma de la **Figura 7.4.**

Mis Grupos

Figura 7.3 Visualización de los títulos

7.1.2. Tablas

Las opciones graficas de las tablas son las siguientes. La clase “mat-header-row” da color de la cabecera de la tabla y centra el texto. La clase “mat-row:hover” permite cambiar el color de la fila donde está encima el ratón y por último la clase “letraCabecera” permite caracterizar el texto que se muestran en la cabecera de la tabla.

```
// Opciones gráficas tablas
.mat-header-row{
  text-align: center;
  background-color: rgb(63, 81, 181);
}

.mat-row:hover {
  background-color: rgb(204, 197, 197);
}

.letraCabecera{
  color: white;
  text-align: center !important;
  font-weight: bold;
  font-size: 90%;
}
```

Figura 7.4 Opciones gráficas tablas

En el componente *.html* se implementa el código de la **Figura 7.5** con el fin de conseguir la tabla deseada (solo se necesita llamar a la clase “letraCabecera” ya que las otras dos clases se cargan automáticamente).

```
<!-- Nombre Column -->
<ng-container matColumnDef="nombre">
  <th mat-header-cell *matHeaderCellDef class="letraCabecera"> Nombre  </th>
  <td mat-cell *matCellDef="let grupo" (click)="EntrarGrupo(grupo)"> {{grupo.Nombre}} </td>
</ng-container>
```

Figura 7.5 Implementación en html de las tablas

Finalmente, la tabla obtiene la forma de la **Figura 7.6.**

Nombre	Descripción
Grupo 1	Física 1ºESO
Grupo 2	Castellano 1º ESO

Figura 7.6 Visualización de las tablas

7.1.3. Botones

Las opciones graficas de los botones son las siguientes. La clase “btn” edita la clase predeterminada de *Angular* con unas características personalizadas. La clase “btn:hover” permite cambiar el color del texto a negro cuando se pone el ratón encima.

```
// Opciones gráficas botones
.btn{
  margin: 5px;
  align-items: center;
  color: white;
  width: 170px;
}
.btn:hover {
  color: black !important;
}
```

Figura 7.7 Opciones gráficas botones

Además, se han personalizado los colores para todos los botones.

```css
// Colores de los botones
.btn.Agregar{
 background-color: rgb(65, 178, 212);
}
.btn.Info{
  background-color: rgb(240, 173, 78);
}
.btn.Back{
  background-color: rgb(240, 173, 78);
}
.btn.Aceptar{
  background-color: rgb(92, 184, 92);
}
.btn.Eliminar{
  background-color: rgb(212, 57, 52);
}
.btn.Editar{
  background-color: rgb(65, 178, 212);
}
```

Figura 7.8 Opciones gráficas botones

En el componente *.html* se implementa el código de la **Figura 7.9** con el fin de conseguir el botón deseado.

```html
<div class="elemento">
  <button type="button" class="btn Editar" (click) = EntrarEditarGrupo(); routerLink= "editarGrupo">
    Editar Grupo <i class="material-icons">edit</i>
  </button>
</div>

<div class="elemento">
  <button type="button" class="btn Eliminar" (click)="AbrirDialogoConfirmacionBorrar()">
    Eliminar Grupo <i class="material-icons">delete</i>
  </button>
</div>
```

Figura 7.9 Implementación en html de los botones

Finalmente, los botones obtienen la forma de la **Figura 7.10.**

Figura 7.10. Visualización de los botones

7.1.4. Cromos

Las opciones graficas de los cromos son las siguientes. La clase “imagenCromo” obliga a la imagen a no sobre pasar unas dimensiones y sobresalir del cromo . La clase “cromo” permite crear las dimensiones del cromo y por último la clase “Diamante” permite caracterizar el cromo que, dependiendo de su nivel, existe una clase para cada uno de los cinco niveles de los cromos.

```
// Opciones gráficas cromos
.imagenCromo {
  max-width: 200px;
  max-height: 150px;
  margin-top: 0px !important;
  margin: 5px;
}
.cromo {
  max-width: 300px;
  max-height: 300px;
  width: 300px;
  height: 300px;
}

.Diamante {
  background-color: rgb(62, 177, 163);
  border-top-left-radius: 15px !important;
  border-top-right-radius: 15px !important;
  border-bottom-left-radius: 15px !important;
  border-bottom-right-radius: 15px !important;
  box-shadow: 0px 3px 6px 0px rgb(50, 150, 138) !important;
}
```

Figura 7.11 Opciones gráficas cromos

En el componente *.html* se implementa el código de la **Figura 7.12** con el fin de conseguir la lista de cromos deseada. Se crea una lista de cromos con la clase “cromo” y el parámetro [ngClass] que permite seleccionar la clase para caracterizar el cromo dependiendo el nivel. Por último, se aplica la clase “imagenCromo” cuando se muestra la imagen.

```
<mat-grid-list cols="3" rowHeight="100px" style="width: 70%; margin-left : 15%" >
  <mat-grid-tile
    *ngFor="let cromo of cromosColeccion; let i = index"
    [colspan]="1"
    [rowspan]="4"
    [style.background]="white">
    <mat-card class="cromo" [ngClass]="{
      'Diamante' : cromo.Nivel === 'Diamante',
      'Platino' : cromo.Nivel === 'Platino',
      'Oro' : cromo.Nivel === 'Oro',
      'Plata' : cromo.Nivel === 'Plata',
      'Bronce' : cromo.Nivel === 'Bronce'}">
      <mat-card-header>
        <div style="display: flex; justify-content:space-between; width: 250px ">
          <div>
            <mat-card-title>{{cromo.Nombre}}</mat-card-title>
            <mat-card-subtitle>
              <p style="margin-bottom: -2px;">Nivel: {{cromo.Nivel}} </p>
              <p style="margin-bottom: -2px;">Probabilidad: {{cromo.Probabilidad}} </p>
            </mat-card-subtitle>
          </div>
          <div class="elementoCromo">

            <button type="button" style="width: 50px !important;" class="btn Editar" routerLink= "editarCromo" (click) = "EnviarCromoEditar(cromo)">
              <i style="margin-top: 3px; margin-right: 8px;" class="material-icons">edit </i>
            </button>

            <button type="button" style="width: 50px !important;" class="btn Eliminar" (click) = "AbrirDialogoConfirmacionBorrarCromo(cromo)">
              <i style="margin-top: 3px; margin-right: 8px;" class="material-icons">delete </i>
            </button>

          </div>
        </div>
      </mat-card-header>
      <mat-card-content style="display: flex; align-items: center;justify-content: center;">
        <div class="posicionImagen" >
          <img class ="imagenCromo" *ngIf="imagenCromoArray[i]" mat-card-image [src]="imagenCromoArray[i]" >
        </div>
      </mat-card-content>
    </mat-card>
  </mat-grid-tile>
```

Figura 7.12 Implementación en html de los cromos

Finalmente, los cromos obtienen la forma de la **Figura 7.13.**

Figura 7.13 Visualización de los cromos

7.1.5. Módulos de creación

Para mantener la coherencia entre estos módulos se ha utilizado un *Stepper* para realizar la creación. Además, el aspecto gráfico es prácticamente el mismo, los botones de cambiar de *Stepper* se sitúan abajo a la derecha y la zona de creación está delimitada por un marco.

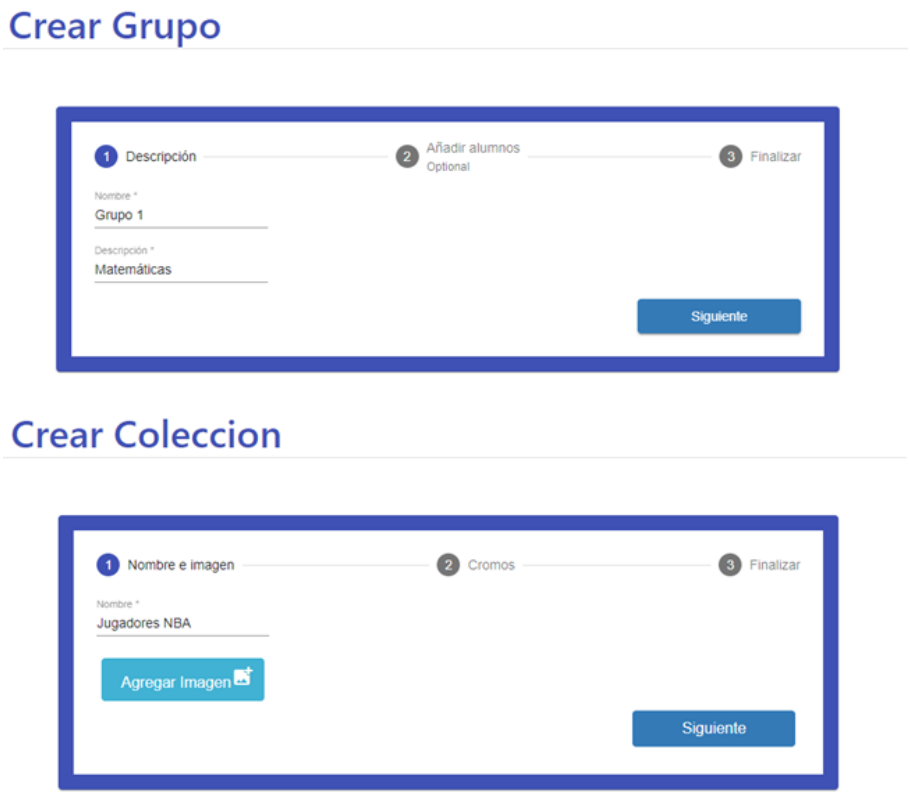


Figura 7.14 Visualización de los módulos de creación

Lista de creaciones

Del mismo modo que ocurre en las ventanas de creación, para las ventanas donde se listan estas creaciones siguen una coherencia como se podrá observar en la **Figura 7.15**. El aspecto gráfico es prácticamente el mismo, las creaciones se muestran con Extension *Panels* unos paneles que al extenderse muestran la información de la creación, en la parte derecha se sitúan dos botones uno para editar y otro para eliminar la creación.

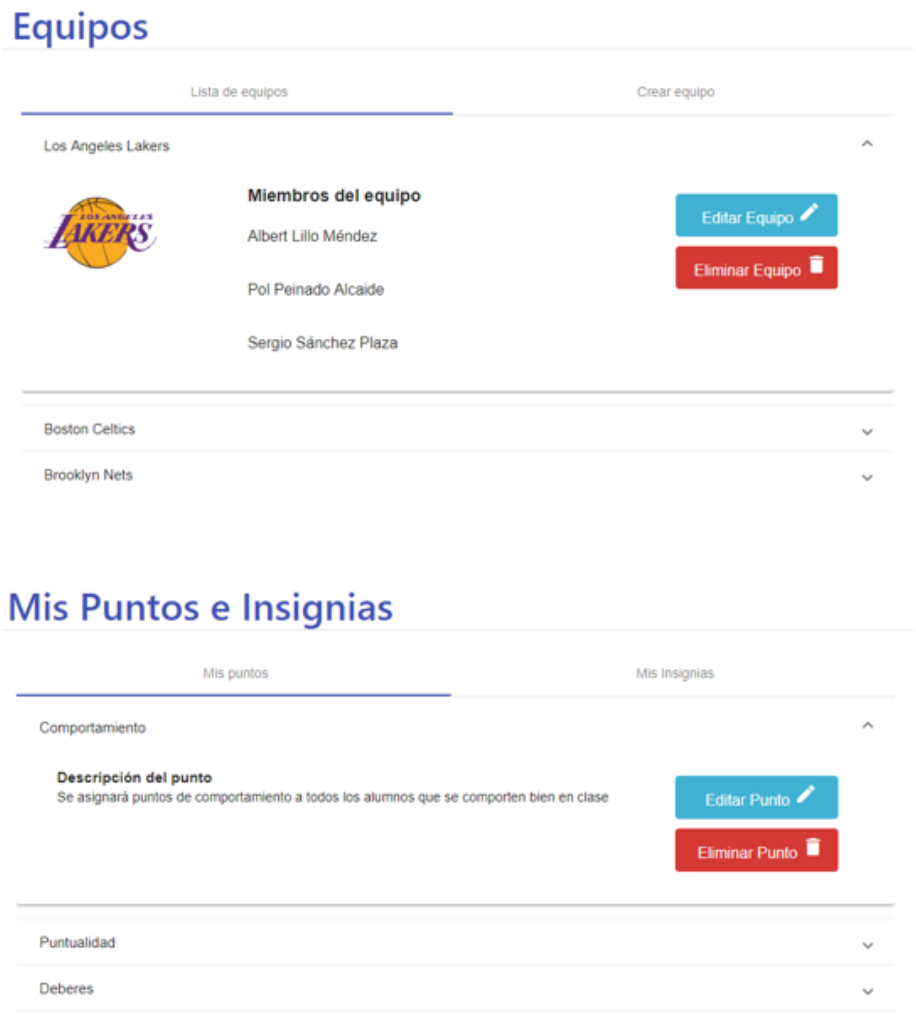


Figura 7.15 Visualización de los módulos de listado de las creaciones

Editar creaciones

A la hora de editar la creación también se mantiene esta coherencia y estructura. Esta estructura se muestra de la siguiente forma, los datos de la creación (nombre, descripción o imagen) se muestran en la parte superior izquierda mientras que el botón para aceptar se sitúa en la parte superior derecha. En la parte inferior se muestra la información de la creación, en estos casos en forma de tabla y con los botones para realizar acciones en esta venta encima de la tabla.

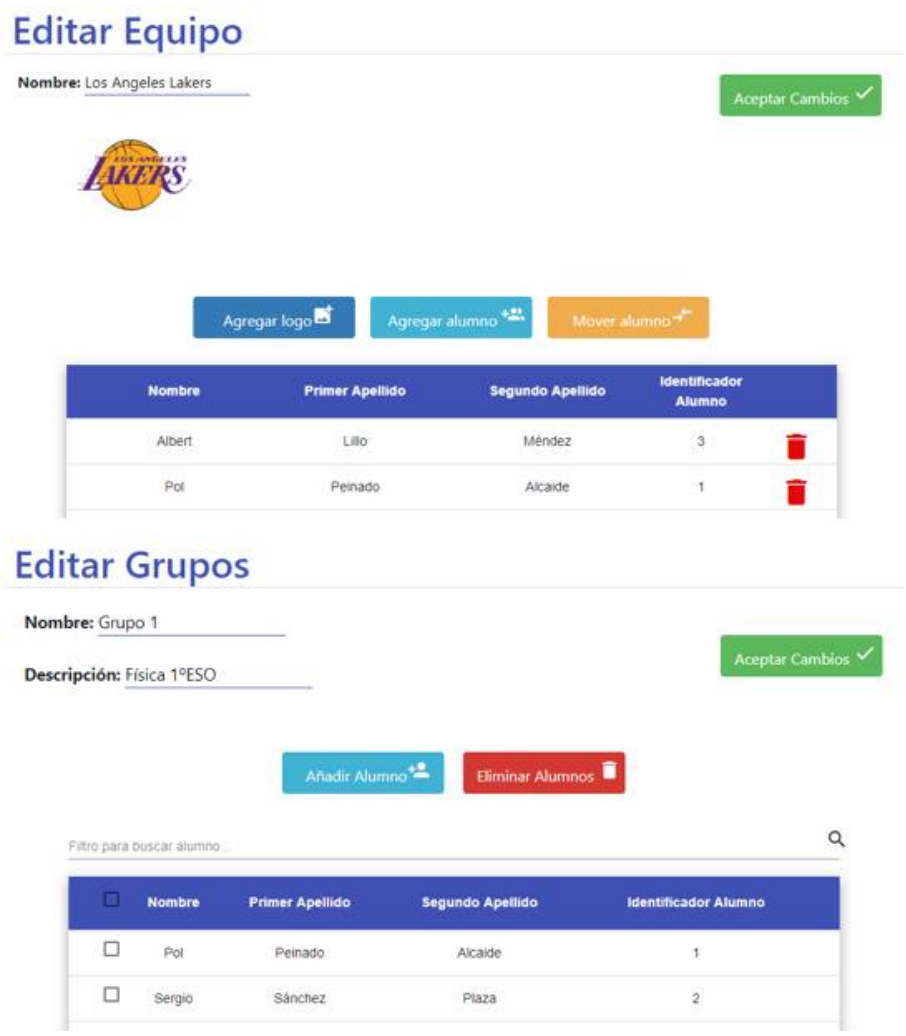


Figura 7.16 Visualización de los módulos de editar de las creaciones

7.1.6. Navbar, footer y diálogo de confirmación

Los componentes navbar y footer mostrados en la **Figura 7.17** permanecen estáticos en todas las pantallas, es decir, son los únicos elementos que se comparten entre todos los componentes. Con la ayuda del navbar se puede navegar por todas las pantallas mientras que con el footer se puede obtener los avisos legales y de privacidad de Classpip.

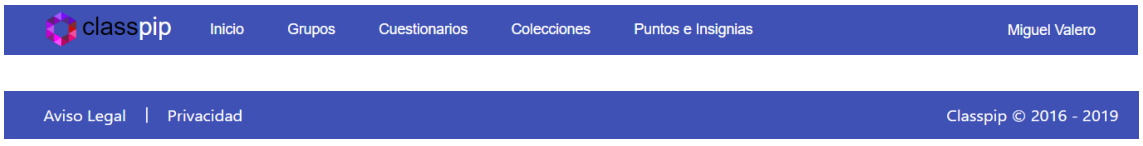


Figura 7.17 Visualización del navbar y footer

El diálogo de confirmación (**Figura 7.18**) permite compartir un mismo componente para todas las acciones de eliminación. Su utilidad viene dada debido a que puede haber un error a la hora de eliminar cierto elemento, por lo que un diálogo de confirmación permite enmendar el error.

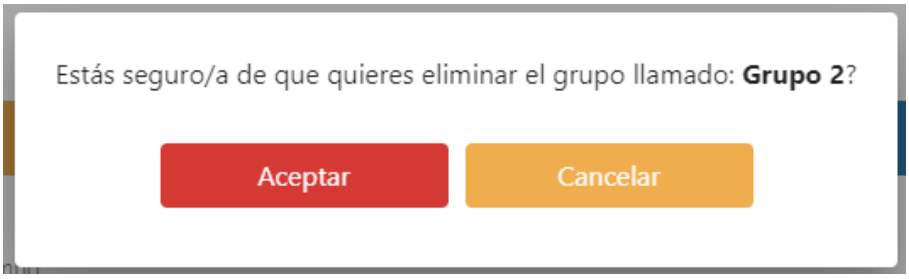


Figura 7.18 Visualización diálogo de confirmación

7.2. Mobile

Para la modalidad Mobile, se buscó seguir el diseño de pantalla comentado en el capítulo 5.2.

A medida que se trabajaba con la estética de las pantallas, se apreció la complejidad de seguir el estilo predefinido en Mockingbot. Esta complejidad es debida a las limitaciones de importación de librerías externa a Ionic. A diferencia de angular, Ionic limita las importaciones de componentes externos por temas de compatibilidad y consecuentemente uno se ve con la necesidad de realizar los componentes de manera manual y sin hacer uso de librerías ya creadas.

Para el estilo gráfico de las pantallas se ha tenido en cuenta la coherencia entre los estilos de los módulos, es decir, para todas las secciones de asignación de puntos o cromos se utiliza un pat*rón de pantalla* y para las secciones donde se informan sobre los elementos a los que optas en un juego se utiliza otro patrón. Además, se ha mantenido la coherencia entre los tipos de letra, tablas y botones. A continuación, se muestran los patrones definidos para el estilo gráfico de Mobile:

7.2.1. Pantallas de la introducción

Al iniciar la aplicación, se generarán unas pantallas, las cuales te introducirán las funcionalidades existentes de Classpip Mobile. Como se aprecia a continuación, todas las pantallas se rigen con el mismo patrón estético y posición de los elementos:

- Enlace en la parte superior derecha de la pantalla para evitar la introducción.
- Imagen relacionada con el texto explicativo que se encuentra en la parte inferior.
- Elementos relevantes de la pantalla de introducción mediante colores azulados.
- Texto informativo sobre el elemento relevante citado posteriormente.

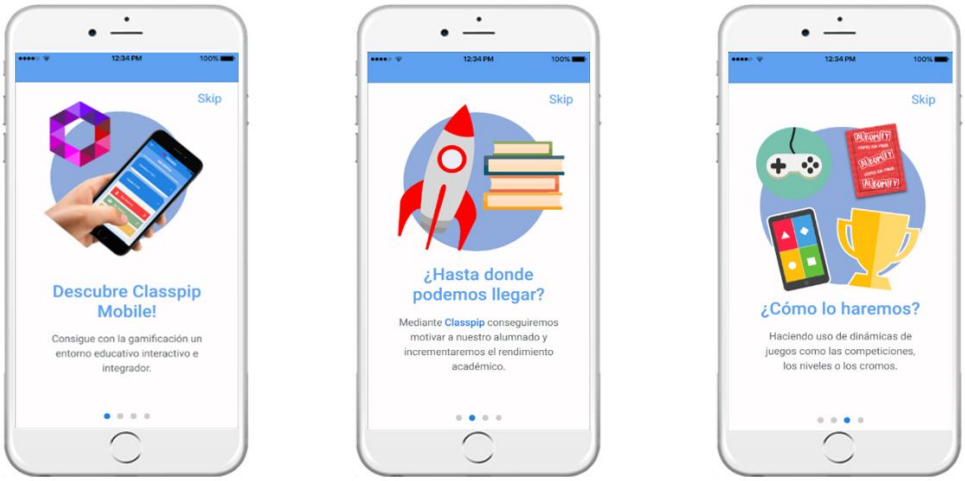


Figura 7.18 Visualización de las pantallas de introducción

7.2.2. Pantalla de acceso a menú lateral

Dicho acceso rápido aparecerá en todas las pantallas existentes excepto en las pantallas deslizables del inicio de la aplicación y en la pantalla de autentificación.

Dicho botón consiste en un acceso rápido al siguiente menú de opciones laterales donde se puede acceder a las pantallas de:

- Mis grupos
- Mis puntos/insignias
- Mis colecciones
- Configuración
- Cerrar Sesión

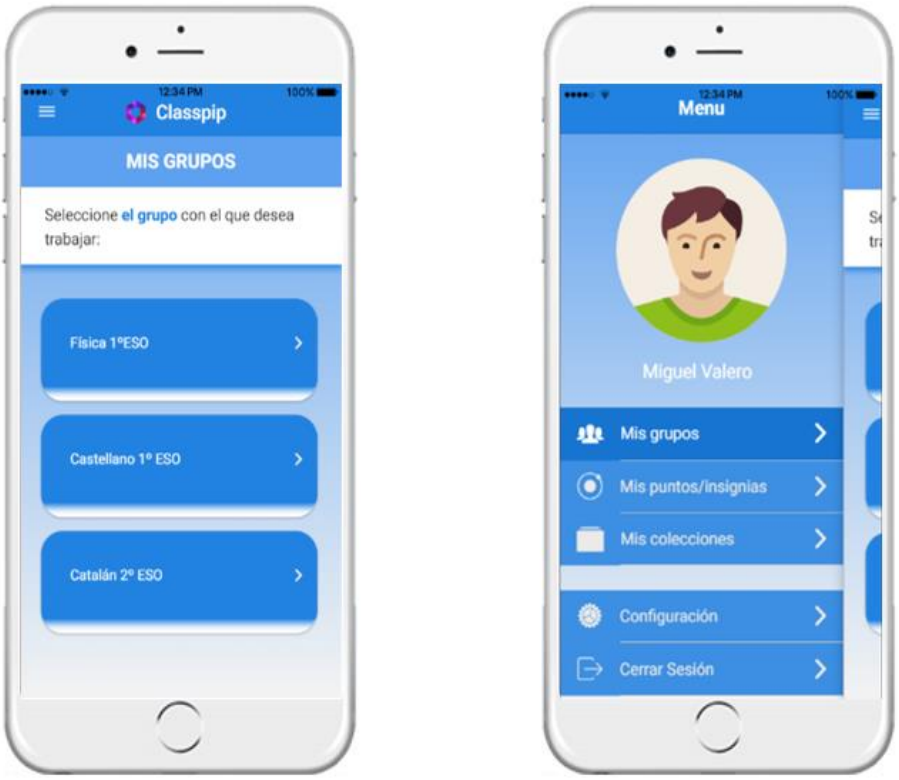


Figura 7.19 Visualización del menú lateral

7.2.3. Pantallas mostrar alumnos de un grupo y de un equipo

Como se puede apreciar en la **Figura 7.20**, las pantallas Alumnos de un Grupo y Alumnos de un Equipo disponen de:

- Un contenedor blanco donde se almacena la información sobre que se puede realizar en la pantalla actual.
- Un buscador de alumnos por Nombre, Primer Apellido o Segundo Apellido.
- Tabla deslizable donde se recogen todos los alumnos existentes.

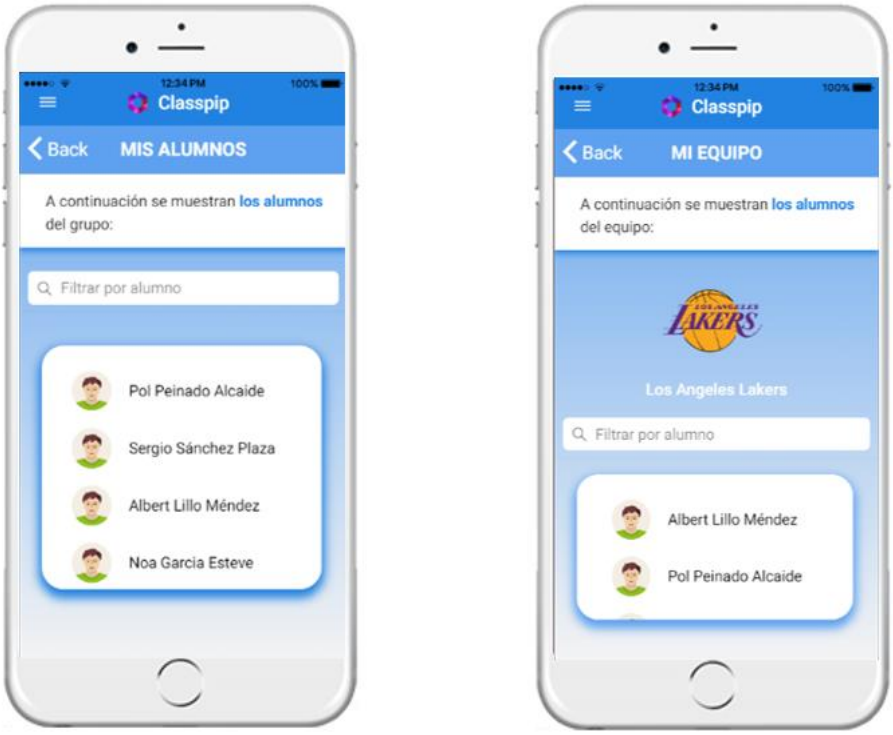


Figura 7.20 Visualización de pantalla alumnos y equipos

7.2.4. Pantallas de asignación de cromos y puntos

Como se puede apreciar en la **Figura 7.21**, las pantallas asignar cromos y asignar puntos disponen de:

- Un contenedor blanco donde se selecciona el tipo de puntos y cantidad de puntos/ Selección del Cromo (Manual)/ cantidad de cromos (Aleatorio).
- Un seleccionador de alumnos.
- Tabla deslizable donde se recogen todos los alumnos existentes.
- Botón de Validación de Asignación.

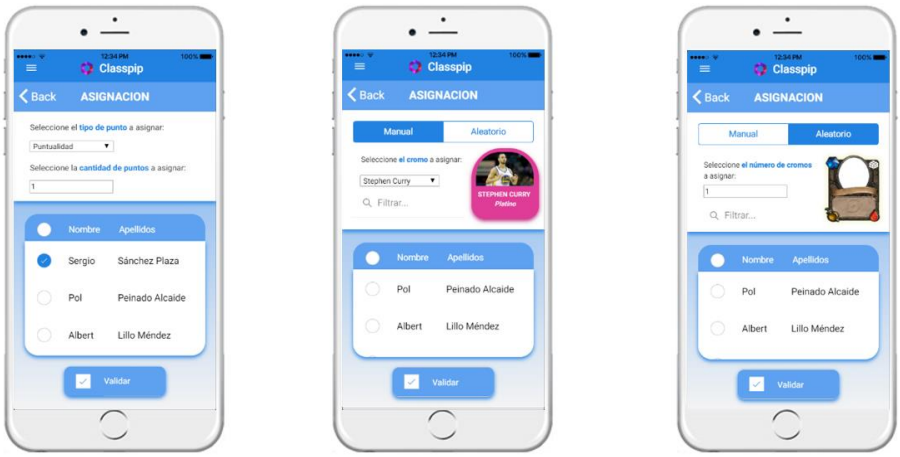


Figura 7.21 Visualización de pantalla asignación

7.2.5. Pantallas de inicio de sesión, alertas y elementos de espera

A continuación, se muestran las alertas, elementos de espera y elementos de inicio de sesión que aparecen al autentificarse:

- Inicialmente aparecerá la pantalla de autentificación de Classpip Mobile.
- Una vez hecho *clic* en Iniciar Sesión, se iniciará el componente de espera hasta que se verifique las credenciales.
- En caso de una validación efectiva, se accederá a la pantalla principal de Classpip Mobile, en caso contrario aparecerá una alerta indicando que se introdujeron incorrectamente las credenciales del usuario.

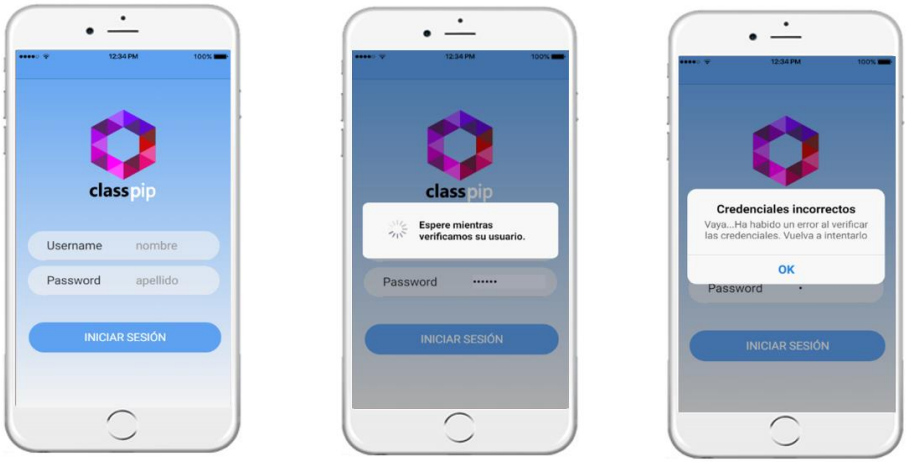


Figura 7.22 Visualización de pantalla inicio sesión, alertas y elementos de espera.

7.2.6. Pantalla de asignaciones y sus consecuentes diálogos de confirmación, alertas y elemento de espera

A continuación, se muestran los elementos de confirmación, alertas y elementos de espera que aparecen al asignar cromos/puntos:

- En los diálogos de confirmación, se indica si la asignación al alumno correspondiente ha sido bien definida o no.

- Posteriormente aparecerá el elemento de espera, permitirá realizar la asignación del cromo y guardarlo en la API.

- Finalmente aparecerá una alerta indicando el estado del procedimiento. En caso de que se haya asignado satisfactoriamente, aparecerá *asignación efectuada*, en caso contrario aparecerá *asignación abortada*.

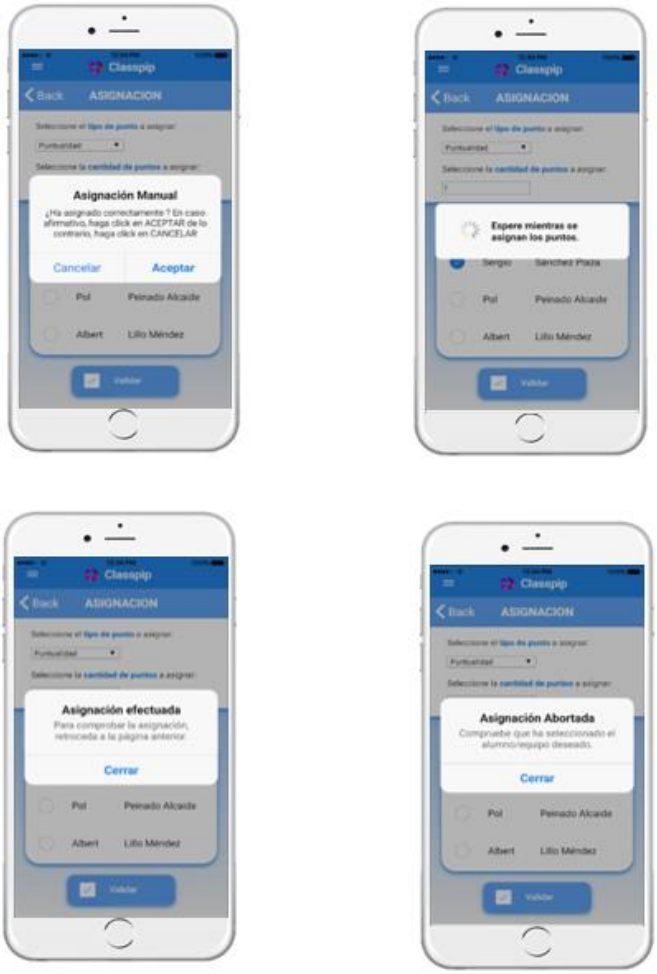


Figura 7.23 Visualización de pantalla inicio sesión, alertas y elementos de espera para pantallas de asignación.

7.2.7. Pantalla de juegos disponibles

Como se puede apreciar en la **Figura 7.24**, la pantalla correspondiente a los juegos disponibles consta de los siguientes elementos:

- Un contenedor blanco donde se selecciona el tipo de juego que se desea visualizar.
- Una tabla con los juegos activos filtrados por el tipo de juego que se ha seleccionado en el desplegable superior.
- Una tabla con los juegos inactivos filtrados por el tipo de juego que se ha seleccionado en el desplegable superior.

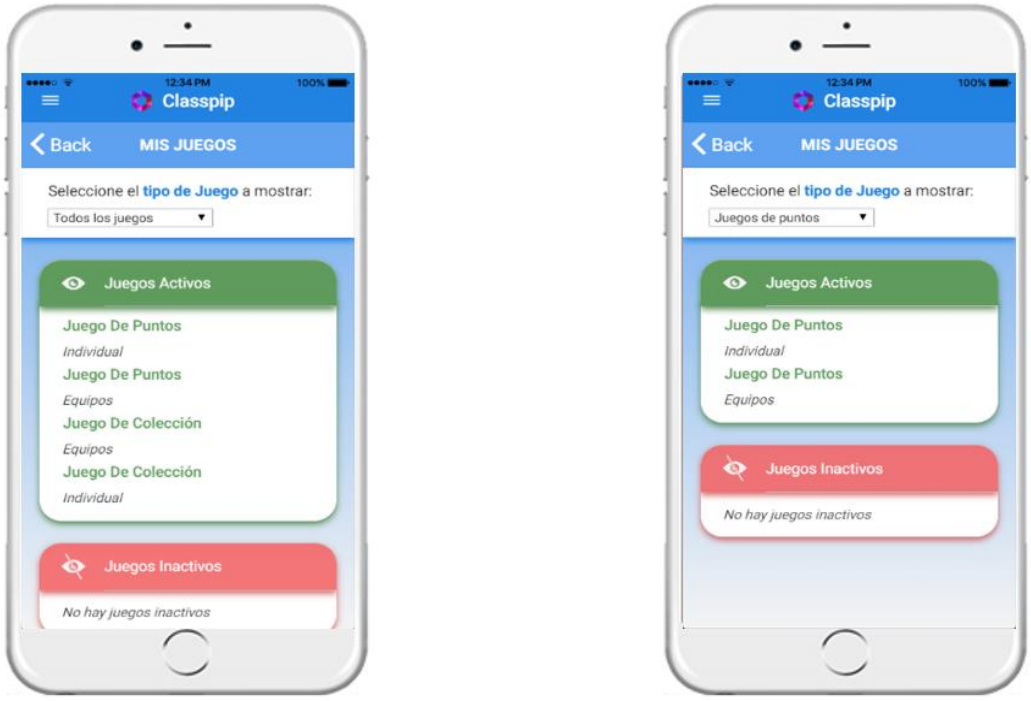


Figura 7.24 Visualización de pantalla de juegos disponibles

7.2.8. Pantalla de cromos de un alumno y cromos de una colección

A continuación, se muestran los elementos existentes en las pantallas referentes a cromos:

- Un contenedor blanco donde se informa qué se va a visualizar en dicha pantalla.

- Dos cromos por fila que en el caso de la pantalla Cromos de un alumno, se diferencian por la transparencia del cromo según si disponemos del cromo o no. Para el caso de cromos de una colección, aparecerán todos con la misma transparencia, pero se diferencian por el color de nivel del propio cromo.

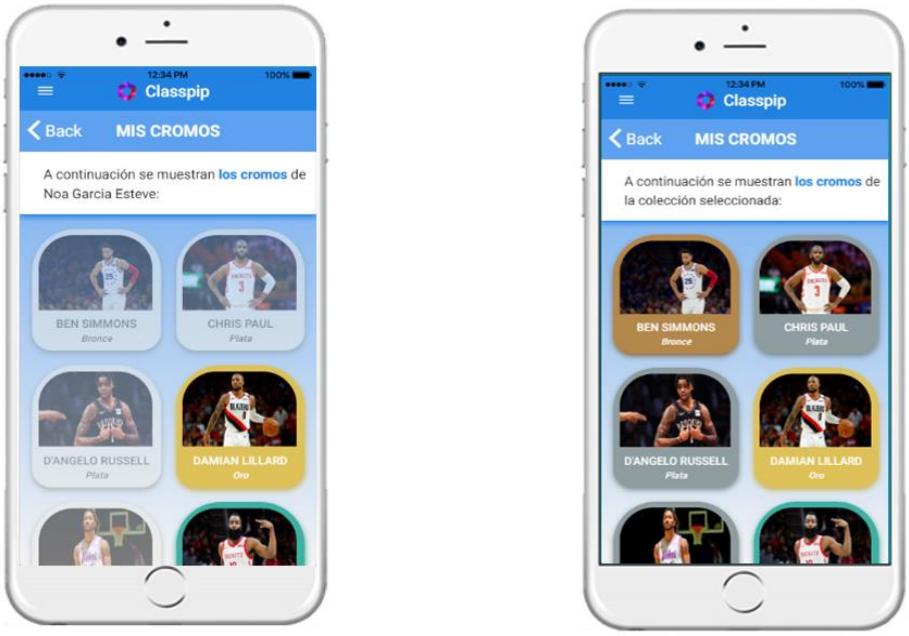


Figura 7.25 Visualización de pantalla de cromos

7.2.9. Pantalla de juego seleccionado

Como se puede apreciar en la **Figura 7.26**, las pantallas correspondientes al Juego Seleccionado, disponen de:

- Un contenedor blanco donde se informa qué se va a visualizar en dicha pantalla.

- Un botón rojo donde te redirige a la información del juego seleccionado. Los puntos a los que se opta, cromos o niveles.

- Un botón verde donde te redirige a la pantalla de asignación de cromos o puntos.

- Un filtrador por tipo de punto o por palabras clave de un alumno (Nombre/ Apellidos).

- Tabla deslizable donde se recoge la información de los alumnos que participan en el juego seleccionado.

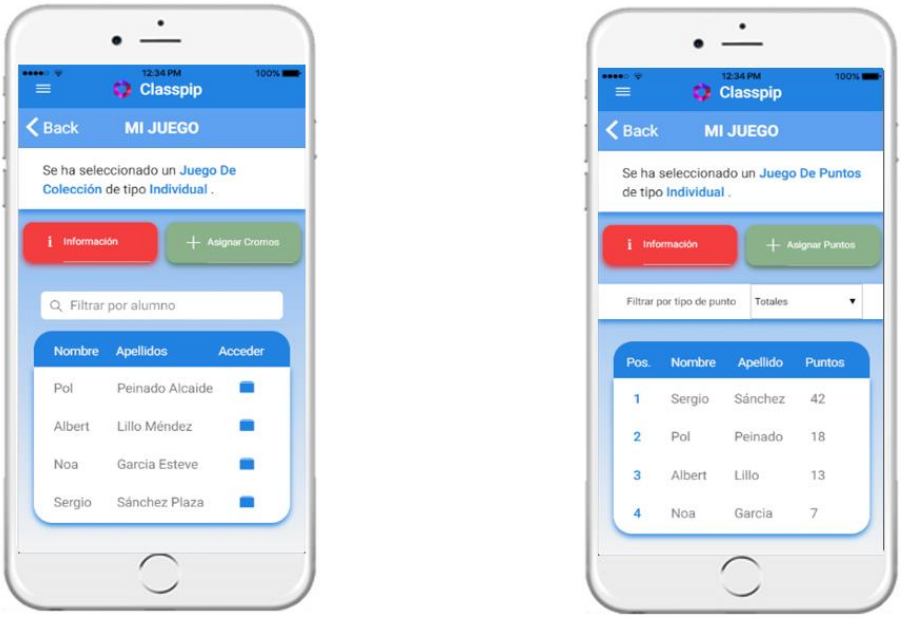


Figura 7.26 Visualización de pantalla de juego seleccionado

7.2.10. Componente Accordion

Como se ha comentado al inicio del capítulo, en Ionic no podemos importar componentes ya definidos desde librerías externas por un problema de compatibilidad de versiones. Para ello ha sido necesario generar un componente llamado Accordion que utilizando elementos ya existentes en Ionic, permitiese mostrar y ocultar información de un elemento al hacer *clic* en él, consiguiendo así una reducción de espacio y una mejora estética.

Figura 7.27 Visualización de pantalla con componente Accordion

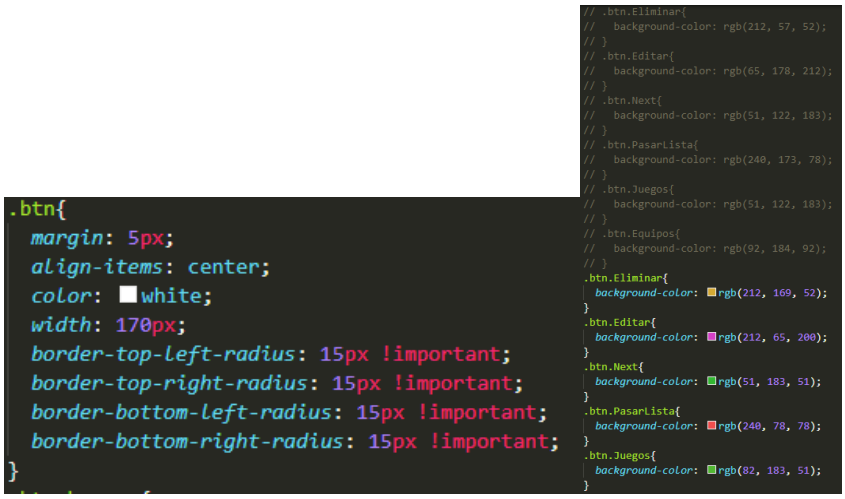
7.3. Facilidad de modificación del estilo gráfico

Una vez explicado la centralización del estilo gráfico en Dashboard, se ha realizar una demostración de cómo podría editarse fácilmente el aspecto de la aplicación. En la **Figura 7.28** se muestra como se edita el color de los títulos, en gris sale comentado el color predeterminado.

```css
.titulo{
  width: 70%;
  margin-left: 15%;
  height: 60px;
  padding: 10px;
  // color: rgb(63, 81, 181);
  color: rgb(181, 87, 63);
}
```

Figura 7.28 Editado del título

En la **Figura 7.29** se muestra como se edita la forma (izquierda) y el color (derecha) de los botones, se le añade un borde más redondeado y en gris sale comentado el color predeterminado.



```css
.btn{
  margin: 5px;
  align-items: center;
  color: white;
  width: 170px;
  border-top-left-radius: 15px !important;
  border-top-right-radius: 15px !important;
  border-bottom-left-radius: 15px !important;
  border-bottom-right-radius: 15px !important;
}
```

```css
// .btn.Eliminar{
//   background-color: rgb(212, 57, 52);
// }
// .btn.Editar{
//   background-color: rgb(65, 178, 212);
// }
// .btn.Next{
//   background-color: rgb(51, 122, 183);
// }
// .btn.PasarLista{
//   background-color: rgb(240, 173, 78);
// }
// .btn.Juegos{
//   background-color: rgb(51, 122, 183);
// }
// .btn.Equipos{
//   background-color: rgb(92, 184, 92);
// }
.btn.Eliminar{
  background-color: rgb(212, 169, 52);
}
.btn.Editar{
  background-color: rgb(212, 65, 200);
}
.btn.Next{
  background-color: rgb(51, 183, 51);
}
.btn.PasarLista{
  background-color: rgb(240, 78, 78);
}
.btn.Juegos{
  background-color: rgb(82, 183, 51);
}
```

Figura 7.29 Editado de los botones

En la **Figura 7.30** se muestra como se edita el color de la cabecera de las tablas, en gris sale comentado el color predeterminado.

```css
.mat-header-row{
  text-align: center;
  // background-color:rgb(63, 81, 181);
  background-color:rgb(63, 181, 79);
}
```

Figura 7.30 Editado de la tabla

El resultado final una vez editados los parámetros explicados previamente se muestra en la **Figura 7.3.** De una forma muy sencilla se han realizado cambios a gran nivel, es decir, se han cambiado los títulos, los botones y las tablas de toda la aplicación.

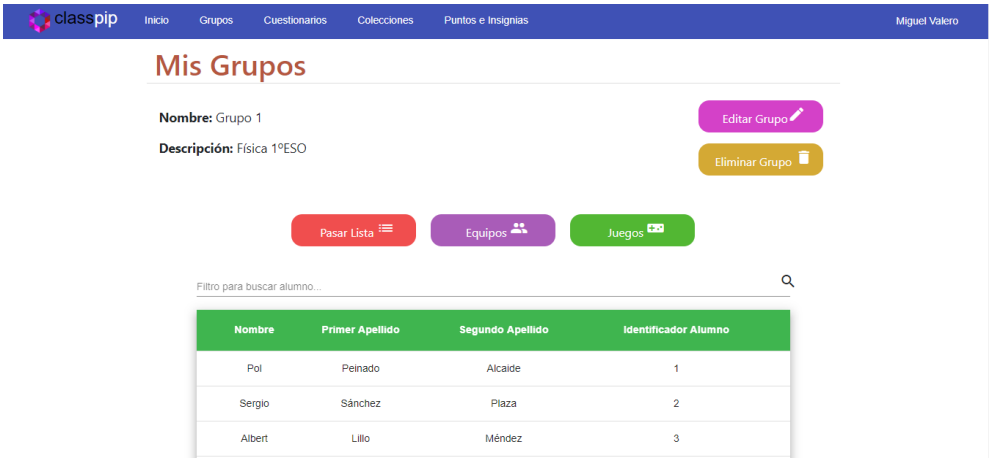


Figura 7.31 Demostración de la centralización de estilos gráficos

CAPÍTULO 8. COMPARACIÓN ENTRE VERSIONES DE CLASSPIP

Al empezar de nuevo todo el proyecto Dashboard, es interesante hacer una comparación en profundidad entre la aplicación de antes y la actual. Para distinguir las versiones, de ahora en adelante las llamaremos Classpip 2018 y Classpip 2019 respectivamente.

La comparación se realizará únicamente con el Dashboard, ya que el Mobile 2018 no estaba apenas desarrollado.

8.1. Inicialización de la aplicación

La aplicación Dashboard funcionará introduciendo la siguiente dirección web en el navegador: http://localhost:4200. Una vez introducida, comentaremos a continuación las páginas que nos salen en cada caso respectivamente.

8.1.1. Inicialización Classpip 2018

En la aplicación del año anterior nos aparece la pantalla que se visualiza a continuación en la **Figura 8.1**.

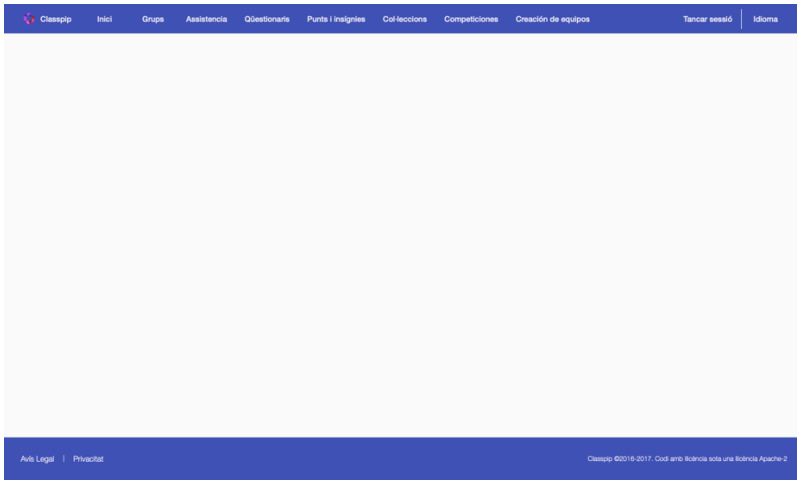


Figura 8.1 Inicialización versión 2018

Como se puede ver, no nos muestra nada de la aplicación ni nos permite acceder a las funcionalidades asociadas a los diferentes botones del navbar. En un mensaje aparece que esto pasa porque no nos hemos autentificado.

Sin embargo, en la esquina superior derecha podemos leer “Tancar sessió”, lo que nos hace pensar que ya estamos dentro la aplicación con una sesión iniciada. Para poder acceder a la aplicación deberemos cerrar sesión y volver a entrar para que nos autentifique.

Al cerrar sesión nos saldrá la pantalla de la **Figura 8.2**.

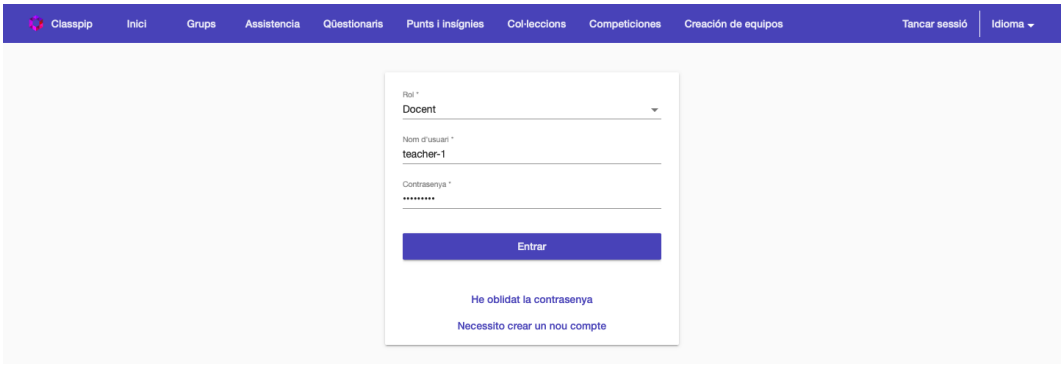


Figura 8.2 Inicio sesión versión 2018

8.1.2. Inicialización Classpip 2019

En la versión actual de Classpip, cuando se inicializa la app nos redirige a la pantalla de iniciar sesión como se puede ver en la **Figura 8.3**, la cual es parecida a la vista anteriormente en la **Figura 8.2**. Por consiguiente, podremos iniciar sesión directamente sin tener que cerrarla previamente.

Figura 8.3 Inicio sesión versión 2019

Otros aspectos importantes a tener en cuenta son el hecho de que no nos salga el navbar si no se ha iniciado sesión y la desaparición de la casilla rol del inicio de sesión.

Dashboard es una herramienta de profesor, con lo que el rol de alumno o escuela desaparecen.

También es importante destacar, como se ha explicado en capítulos anteriores, que la nueva versión de Classpip no ha utilizado la base User en el modelo Profesor para facilitar el trabajo. Con lo que una de las futuras mejoras sería incluir ese modelo y llevar a cabo la Lección de Autentificación Autentica.

También se han tenido dificultades con el footer de la nueva versión, ya que éste no se queda fijo independientemente del tamaño del contenido, sino que se adapta. Con lo que a partir de ahora se podrá visualizar que el footer va cambiando su posición. Éste es otro punto a mejorar en futuras versiones.

8.2. Inicio y navbar

La pantalla de inicio es la primera que aparece una vez hemos iniciado sesión. En ella deberá aparecer información relevante para el profesor. Además, deberá ser visible la barra de navegación para que el profesor pueda acceder rápidamente a las funciones más importantes.

8.2.1. Inicio y navbar Classpip 2018

En la pantalla de inicio se visualiza lo mismo que en la pantalla Classpip: un texto modelo y el ranking de los estudiantes.

La barra de navegación cuenta con botones no desplegables que crean accesos directos a las diferentes pantallas.

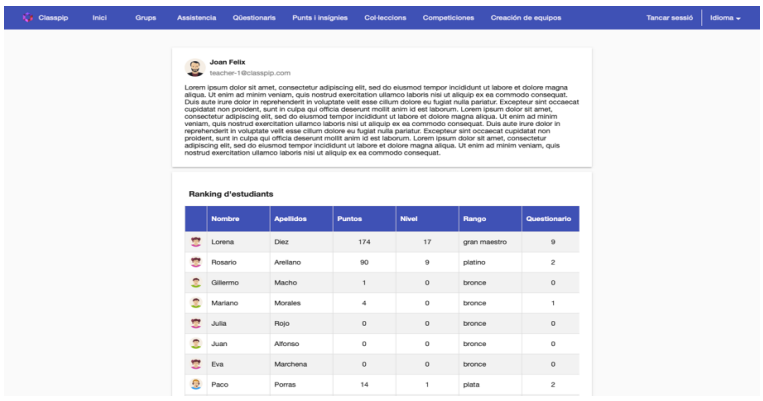


Figura 8.4 Inicio y navbar versión 2018

8.2.2. Inicio y navbar Classpip 2019

Nuestro inicio consta de una serie de botones que al hacer clic encima nos abre una pestaña nueva con video tutoriales de las diferentes opciones del navbar.

Con esto se pretende dar información útil a los profesores que utilicen Classpip por primera vez. Se optó por esta opción ya que no veíamos útil poner un texto explicativo y un ranking de alumnos los cuales no especifica el grupo a los que pertenecen.

El navbar está formado de botones desplegables (excepto el Inicio y Classpip) para separar entre las funciones de crear y visualizar el contenido que ya tengo creado. La inclusión de botones desplegables optimizará mucho la aplicación como se podrá ver más adelante.

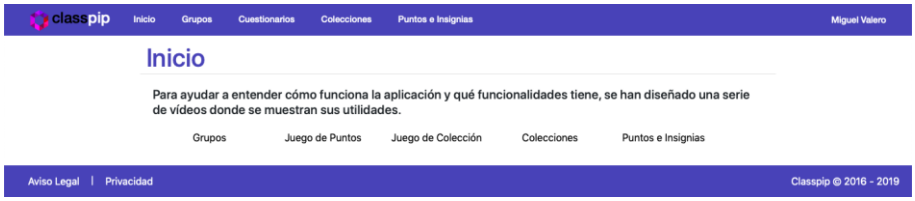


Figura 8.5 Inicio y navbar versión 2019

De ahora en adelante se comparará ambas aplicaciones siguiendo la estructura del navbar del Classpip 2019. Esto se hace porque hay funciones del navbar del Classpip 2018 que están incluidas como subfunciones del de 2019.

Por ejemplo, la creación de equipos que es visible en la barra de 2018 es una subfunción de Grupos visible en 2019.

8.3. Grupos

La funcionalidad de grupos deberá permitir al profesor crear nuevos grupos y realizar diferentes funciones.

8.3.1. Grupos Classpip 2018

Al clicar en grupos nos mostrará una lista con los grupos del profesor. Al entrar en uno de ellos, únicamente nos muestra una lista con los alumnos que forman parte de él.

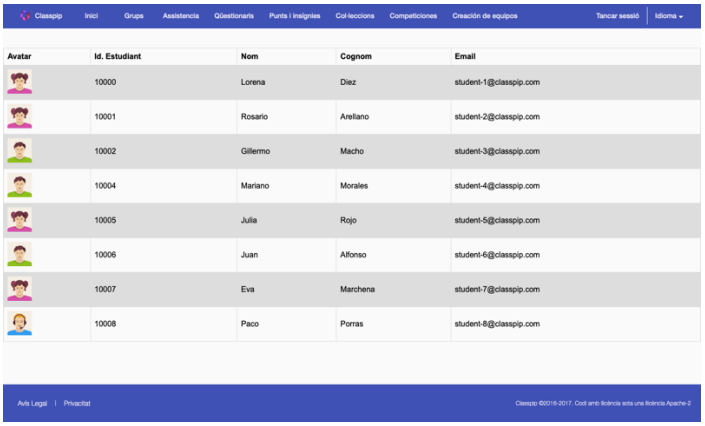


Figura 8.6 Grupos versión 2018

8.3.2. Grupos Classpip 2019

Al clicar en grupos se nos abrirá un desplegable con las opciones “Crear Grupo” y “Mis Grupos”. La opción “Mis Grupos” vuelve a ser una lista con los grupos del profesor. Al acceder a uno de ellos, además de la lista de los alumnos, nos aparecerán las diferentes funciones de dicho grupo. En la **Figura 8.7** se muestran ambas pantallas respectivamente.

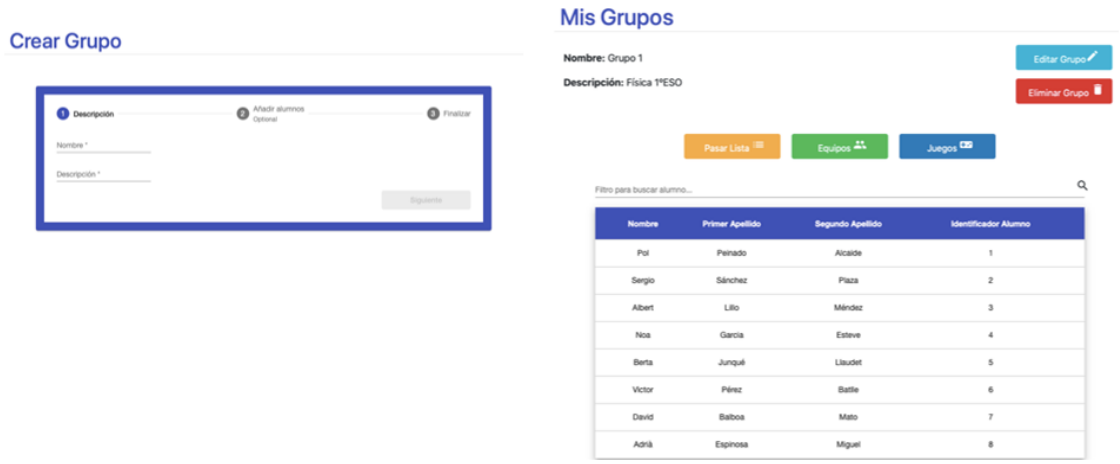


Figura 8.7 Crear grupo (izda.) y mis grupos (dcha.) versión 2019

Crear un grupo resulta muy fácil e intuitivo gracias a un *stepper*. Solo debemos seguir los pasos indicados. En la versión antigua el profesor no tenía forma de crear un grupo.

Dentro de un grupo podemos editarlo (cambiar nombre, descripción, añadir/eliminar alumnos) y eliminarlo.

En la versión actual no se desarrolló la función “Pasar Lista”, con lo que, continuaremos la comparación centrándonos en los equipos y los juegos.

8.3.3. Equipos Classpip 2018

En la versión antigua, la función de equipos, pese a estar vinculada con grupos, se encuentra fuera de ésta. Para acceder a esta función deberemos clicar sobre “Creación de equipos” del navbar.

Pese a que, en una primera instancia, puede parecer que la función únicamente permite crear equipos, al acceder a ella nos damos cuenta que encontramos también la función de visualizar y eliminar. En la **Figura 8.8** se puede ver la inclusión de las tres funciones.

Es destacable que, al estar fuera de grupos, debemos seleccionar en que grupo deseamos crear un equipo y visualizar los que ya hay creados.

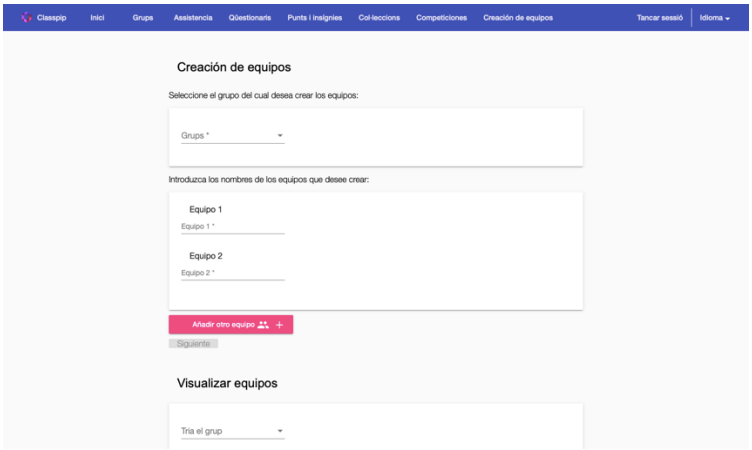


Figura 8.8 Equipos versión 2018

8.3.4. Equipos Classpip 2019

En contraposición con la versión antigua, reorganizamos la manera en como mostrar la información para que aparezca más ordenada y clara. Utilizamos un *tab* que nos permite visualizar/editar los equipos y otro para crearlos. En la **Figura 8.9** se muestran ambos *tabs* respectivamente.

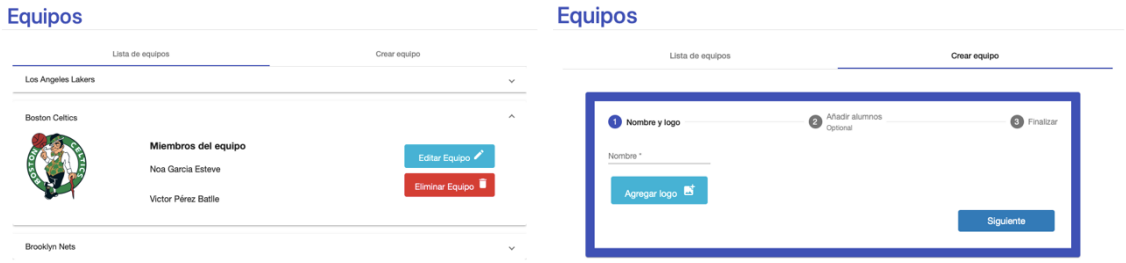


Figura 8.9 Mis equipos (izda.) y crear equipo (dcha.) versión 2019

El hecho de incluir los equipos dentro de grupo es mucho más óptimo, ya que los equipos se crearán/eliminarán directamente en ese grupo, sin necesidad de especificar el grupo como en la versión antigua. Además, solo nos permitirá escoger alumnos de ese grupo para formar los equipos.

Otras funcionalidades nuevas incluyen que el profesor podrá editar el equipo cambiándole el nombre, el logo y los alumnos. En la **Figura 8.10** se muestra como el profesor puede añadir un alumno que todavía no tiene equipo o como puede mover un alumno a otro equipo.

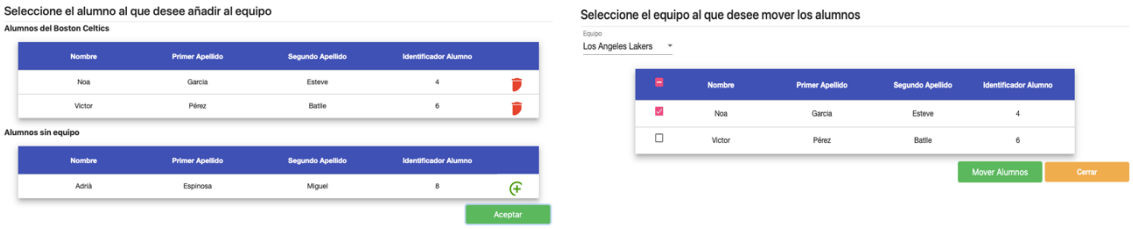


Figura 8.10 Agregar alumnos (izda.) y mover alumnos (dcha.) versión 2019

8.3.5. Juegos Classpip 2018

En la versión de 2018 propone una manera diferente de realizar los juegos. El profesor puede asignar puntos, insignias y cromos a los alumnos/equipos de un grupo determinado, pero no forman parte de ningún juego. Solo permite crear juegos de competición, el cual no se hará la comparación ya que en este

proyecto no se ha llevado a cabo. Con lo que se explicarán los puntos e insignias y las colecciones directamente.

Puntos e insignias Classpip 2018

Como no se ha creado un juego de puntos previamente, el profesor deberá dirigirse a la función “Punts i Insígnies” de la barra de navegación para poder asignarlas.

Nuevamente, al igual que en el caso de equipos, en una misma pantalla muestra la opción crear, asignar, visualizar y eliminar un punto/insignia.

Además, al no estar incluido dentro de grupo, deberá especificar el grupo en el que se encuentra el alumno/equipo al que desea asignar puntos/insignias o del que se desea mostrar la puntuación.

En caso de querer eliminar o asignar un punto/insignia deberemos introducir su ID, lo cual es muy ineficaz.

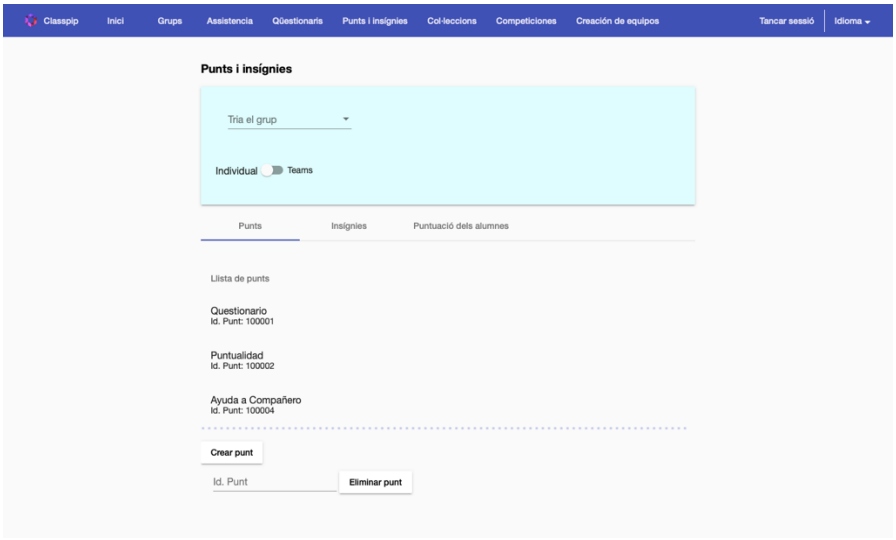


Figura 8.11 Puntos e insignias versión 2018

Colecciones Classpip 2018

Al igual que pasa con el juego de puntos, no podemos crear un juego de colección previamente. Por consiguiente, nos dirigimos al apartado “Col·leccions”.

Una vez más encontramos las opciones asignar, crear y eliminar colección en una única pantalla, de la cual debemos indicar el grupo. En caso de querer eliminar una colección, deberemos introducir su ID.

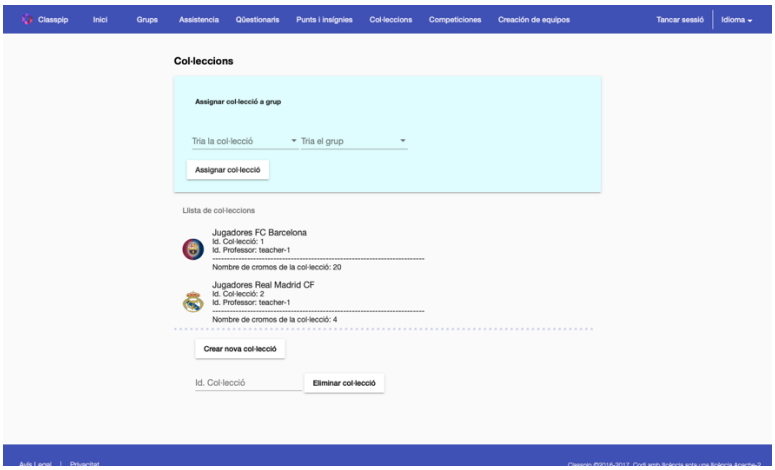


Figura 8.12 Colecciones versión 2018

Si queremos asignar cromos, deberemos entrar dentro de la colección y seleccionar el grupo al que pertenece el estudiante y después buscarlo. Esta pantalla también incluye la opción de añadir y eliminar cromo (indicando su ID).

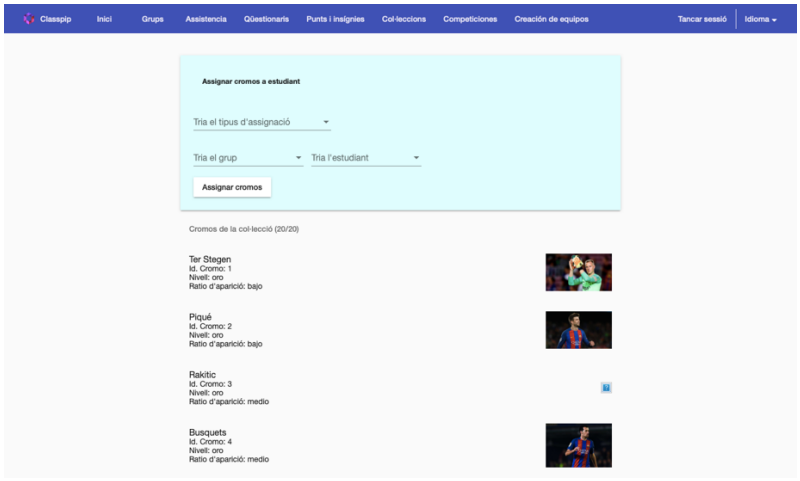


Figura 8.13 Asignar cromo versión 2018

8.3.6. Juegos Classpip 2019

En la nueva versión del proyecto, por el contrario, se pueden crear los diferentes tipos de juego y ver la progresión de los participantes en ellos. El hecho de añadir esta función, nos permite diferenciar entre las pantallas donde creamos los puntos, insignias y colecciones de las pantallas que nos permiten asignarlas.

Además, se ha introducido el concepto de juego activo y no activo. Esto se hace para que los juegos que ya se hayan finalizado aparezcan en otra lista separados de los que aún se están jugando, pudiendo consultar los resultados que se obtuvieron o eliminarlo.

Juego de puntos Classpip 2019

Al seleccionar un juego de puntos veremos las diferentes funcionalidades de la que dispone el juego, así como clasificación de los participantes por el total de puntos, como se puede apreciar en la **Figura 8.14**. No obstante, se puede aplicar un filtro para ver la puntuación por los diferentes tipos de puntos asignados al juego.

En el caso de un juego por equipos, además, podremos ver los miembros de cada equipo.

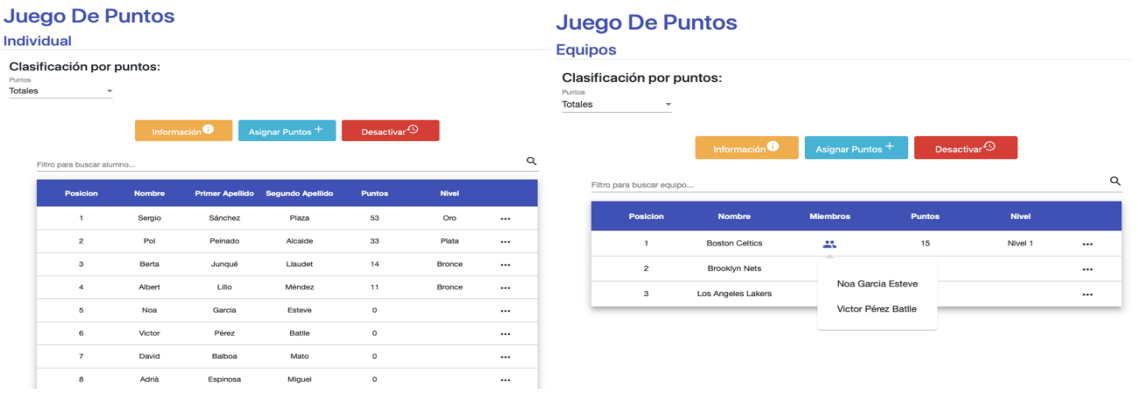


Figura 8.14 Juego de puntos individual (izda.) y equipos (dcha.) versión 2019

Al crear un juego de puntos, el profesor debe escoger los puntos para ese juego de entre todos los que tiene creados. Además, tendrá la opción de crear niveles que los alumnos deberán alcanzar. La función de información nos mostrará los niveles creados y los puntos seleccionados. En la **Figura 8.15** pueden verse sus *tabs* asociados.

Figura 8.15 Información niveles (izda.) y puntos (dcha.) versión 2019

La manera de asignar puntos además es mucho más fácil e intuitiva que en la versión anterior. Al estar dentro de un grupo, nos saldrá directamente la lista de los alumnos a los que podemos asignar puntos. Los puntos que podremos asignar serán únicamente los que escogimos para el juego, y el valor de éstos los decidira el profesor a la hora de asignarlos como se puede ver en la **Figura 8.16**.

Figura 8.16 Asignar punto versión 2019

Por último, el profesor podrá ver el progreso de cada jugador en el juego. Para ello, debe clicar en el icono de los tres puntos del ranking del juego de puntos. El profesor podrá ver el nombre del alumno, los puntos que lleva, su posición, el nivel y una barra de progreso de nivel.

Asimismo, podrá visualizar un historial de los puntos que ha repartido al alumno, con la fecha y el valor del punto. Si algún punto lo asignó por error, tiene la opción de eliminar los puntos del alumno.

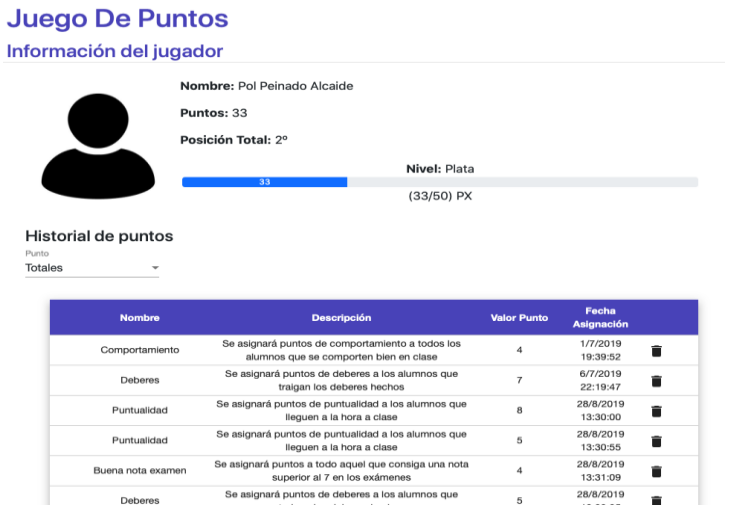


Figura 8.17 Información alumno versión 2019

Juego de colección Classpip 2019

El juego de colección está estructurado de igual manera que el juego de puntos. Habrá un botón que nos de información del juego y otro que nos permita asignar cromos. Al crear el juego le asociamos una colección, a diferencia de la versión antigua que la asignaba a un grupo.

Por lo que, la información del juego que tendremos será la colección seleccionada, así como sus cromos.

Para asignar cromos podremos hacerlo de dos maneras: manual o aleatoriamente. Si elegimos la opción manual, seleccionaremos un cromo especifico de la colección, mientras que si lo hacemos aleatoriamente deberemos indicar el número de cromos que deseamos repartir. En la **Figura 8.18** vemos como repartir cromos de ambas maneras respectivamente.

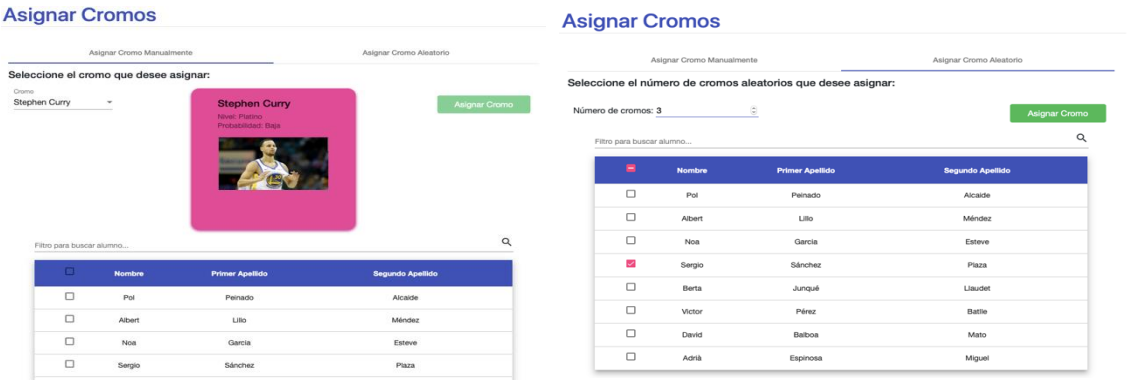


Figura 8.18 Asignar cromo manual (izda.) y aleatorio (dcha.) versión 2019

Al igual que en el caso de juego de puntos, podemos ver el progreso de los participantes en el juego. En el juego de colección esto significa ver los cromos que tengo y los que me faltan de la colección. Deberemos volver a hacer clic en el icono de los tres puntos de la lista y visualizaremos los cromos totales que tiene el participante. Puede ser que tenga cromos repetidos.

Si queremos ver cuantos cromos de la colección tiene, deberemos hacer clic en el botón “Álbum Completo”, donde podremos ver destacados los cromos de los que dispone.

Una de las posibles mejoras en un futuro es incluir la función de intercambiar cromos entre los alumnos. Esta función no se ha incluido actualmente ya que los proyectos desarrollados son para el profesor, con lo que resultaría inútil incluirla porque el intercambio entre cromos debe ser directo entre alumnos, sin interferir el profesor.

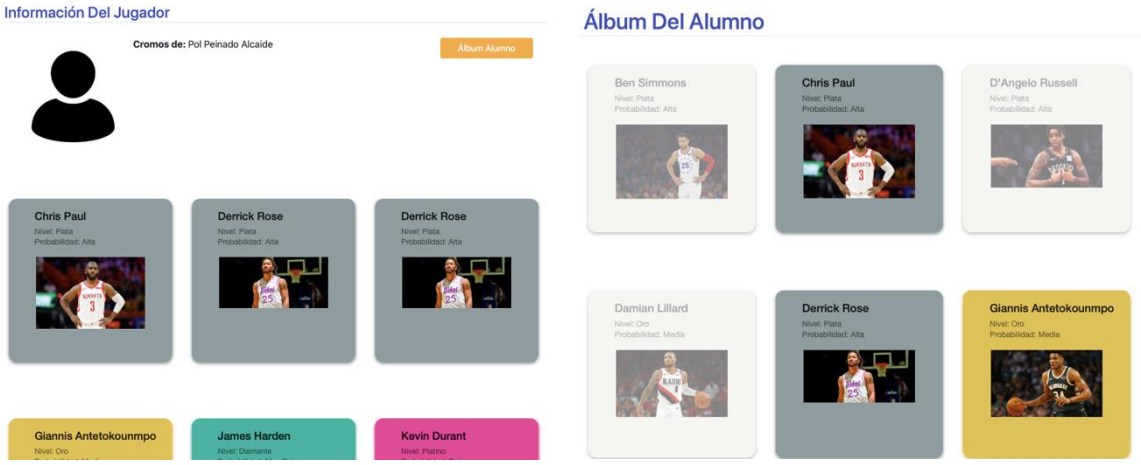


Figura 8.19 Cromos (izda.) y álbum (dcha.) del alumno versión 2019

8.4. Colecciones

En cuanto a las colecciones, debemos recordar que en el apartado de Colecciones Classpip 2018 (dentro de Juegos Classpip 2018) incluye tanto el asignar como todo lo relevante a colecciones. Es por eso que en este apartado explicaremos las colecciones de la nueva versión, ya que la versión anterior ya fue explicada.

8.4.1. Colecciones Classpip 2019

Volvemos a separar entre la opción de crear y visualizar las que ya están creadas. Para eliminar una colección no deberemos introducir el ID, solo clicar a un botón.

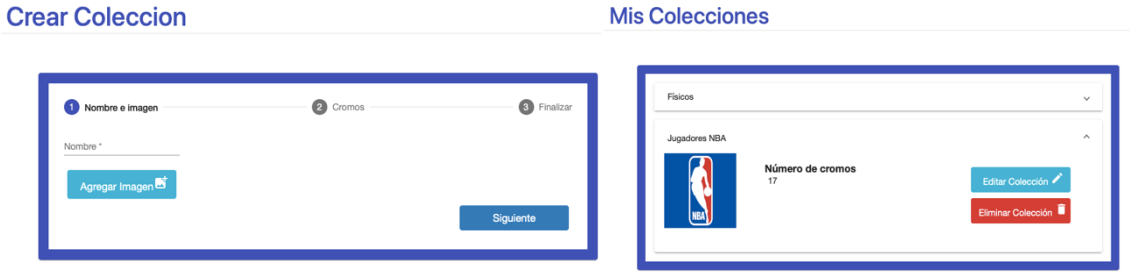


Figura 8.20 Crear colección (izda.) y mis colecciones (dcha.) versión 2019

También podremos editar la colección, cambiando el nombre, la imagen, añadir nuevos cromos o editar/eliminar los que ya existen. A continuación, se muestra en la **Figura 8.21** la función de editar una colección.

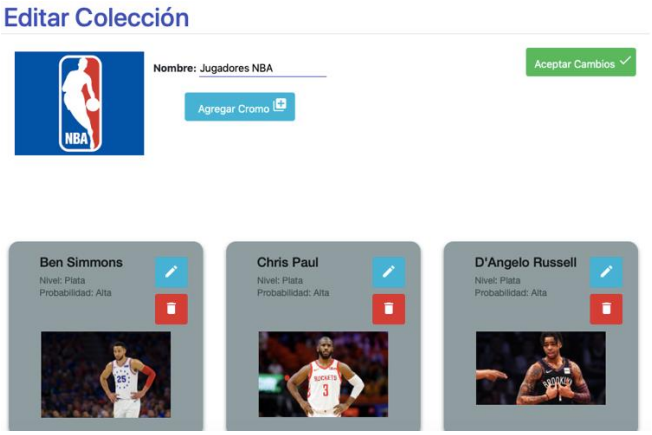


Figura 8.21 Editar colección versión 2019

8.5. Puntos e insignias

Al igual que en el apartado anterior, explicaremos esta función para la versión 2019, ya que para la versión antigua se explicó en el apartado Puntos e Insignias Classpip 2018 dentro de Juegos Classpip 2018.

8.5.1. Puntos e insignias Classpip 2019

Volvemos a separar entre la opción de crear y visualizar. Además, en este caso también separamos entre puntos e insignias con un *tab* en ambas funciones. A continuación, se muestra en la **Figura 8.22** la pantalla de "Crear Puntos e Insignias" en el *tab* de "Crear puntos" y la pantalla de "Mis Puntos e Insignias" en el *tab* de "Mis puntos". En la **Figura 8.23** se muestra lo propio con las insignias.

Figura 8.22 Crear puntos (izda.) y mis puntos (dcha.) versión 2019

Figura 8.23 Crear insignia (izda.) y mis insignias (dcha.) versión 2019

CAPÍTULO 9. IMPLEMENTACIÓN

El estilo de arquitectura de software seguido en Dashboard consiste en el Modelo-Vista-Controlador (MVC), que separa los datos de la aplicación (modelo), la representación visual de los datos (vista) y la lógica de control (controlador).

A continuación, se muestra una distinción según modalidad Dashboard o Mobile, para así conseguir diferenciar las distintas implementaciones utilizadas.

9.1. Implementación Dashboard

Para el desarrollo de la aplicación Dashboard se han añadido modelos correspondientes a las clases implicadas en la aplicación, componentes, los cuales contienen la lógica para interactuar con el usuario y forman las vistas de la representación visual de la interfaz, y servicios, que proveen las funcionalidades a los componentes.

9.1.1. Clases

Los modelos representan la información que maneja la aplicación. En ellos se guarda la información que proviene de la base de datos y permite crear nuevos objetos. Para que la aplicación interprete los modelos, debemos crear las clases.

Estas clases son creadas en TypeScript y deben tener las mismas propiedades que los modelos (los nombres deben coincidir). Además, contienen el constructor del objeto y los métodos para obtenerlo y modificarlo. Podremos encontrar las clases creadas en el proyecto Dashboard en la carpeta ***app*** dentro de ***clases***. Los archivos tienen la extensión *ts*.

A lo largo del trabajo se han creado un total de 27 clases, como por ejemplo la clase Profesor o Alumno. La diferencia de número entre las clases y los modelos se deben a que a priori creamos modelos en el proyecto Service que al final no hemos implementado. No obstante, no se borraron para que en futuros proyectos se utilice.

Como las clases deben tener las mismas propiedades que los modelos del proyecto Service, no se incluirán en el anexo.

```typescript
export class Alumno {
  Nombre: string;
  PrimerApellido: string;
  SegundoApellido: string;
  ImagenPerfil: string;
  profesorId: string;
  id: number;

  constructor(nombre?: string, primerApellido?: string, segundoApellido?: string, imagenPerfil?: string) {

    this.Nombre = nombre;
    this.PrimerApellido = primerApellido;
    this.SegundoApellido = segundoApellido;
    this.ImagenPerfil = imagenPerfil;
  }
}
```

Figura 9.1 Clase Alumno

9.1.2. Componentes

Los componentes proveen las vistas con las que interactuará el usuario, consta de tres tipos de archivos. Uno de los tres archivos es un controlador, contiene los datos y la lógica del componente, este controlador se asocia con un archivo HTML, que crea la arquitectura y da forma a la interfaz gráfica. Además, se ha creado un archivo tipo *README*, que proporciona una breve descripción del controlador, con el fin de facilitar la implementación en el proyecto.

En la **Figura 9.2** se muestra un ejemplo de un componente con su archivo HTML (primera captura), TL (segunda captura) y su fichero explicativo (tercera captura).

```html
<div class="titulo"><h2>Crear Coleccion</h2></div>
<mat-divider style="width: 70%; margin-left : 15%"></mat-divider>

<div class="contenedor">
  <!-- STEPPER -->

    <mat-card class="panel">

      <mat-card-content>

      <mat-horizontal-stepper #linearHorizontalStepper="matHorizontalStepper" [linear]="true" #stepper>


        <!-- PRIMER PASO -->
        <mat-step [stepControl]="myForm">

        <form [formGroup]="myForm">
          <ng-template matStepLabel>Nombre e imagen</ng-template>
          <mat-form-field>
            <mat-label>Nombre</mat-label>
            <input matInput formControlName="nombreColeccion" required>
            <mat-error>Este campo es obligatorio</mat-error>
          </mat-form-field>
```

```
// Creamos una coleccion dandole un nombre y una imagen
CrearColeccion() {

  let nombreColeccion: string;

  nombreColeccion = this.myForm.value.nombreColeccion;

  console.log('Entro a crear la coleccion ' + nombreColeccion);
  console.log(this.nombreImagen);

  // Hace el POST del equipo
  this.coleccionService.POST_Coleccion(new Coleccion(nombreColeccion, this.nombreImagen), this.profesorId)
  .subscribe((res) => {
    if (res != null) {
      console.log(res);
      this.coleccionYaCreada = true; // Si tiro atrás y cambio algo se hará un PUT y no otro POST
      this.coleccionCreada = res; // Lo metemos en coleccionCreada, y no en coleccion!!

      // Hago el POST de la imagen SOLO si hay algo cargado. Ese boolean se cambiará en la función ExaminarImagen
      if (this.imagenCargado === true) {

        // Hacemos el POST de la nueva imagen en la base de datos recogida de la función ExaminarImagen
        const formData: FormData = new FormData();
        formData.append(this.nombreImagen, this.file);
        this.coleccionService.POST_ImagenColeccion(formData)
```

```
# OBJETIVO:  dirigir al profesor de la manera más clara y sencilla posible en el proceso de creación. Como la creación
consta de varios pasos, nos hemos decantado por un stepper. Crearemos la coleecion y añadiremos los cromos de manera muy
simple.


<!-- crear-coleccion.component.ts -->

Al iniciar el crear-coleccion recuperaremos el profesorId de la URL, el cual será un string. Para convertir un string en
number utilizamos el método Number, por defecto en Visual Code.

También iniciamos la URLVueltaInicio, la cual nos enviará a la pantalla de inicio una vez acabado de crear la colección.

Los formGruop los usamos para el stepper del html. Tendremos un stepper con tres pasos. Recogeremos información del primero,
así que crearemos un formGroup (myForm) el cual tendrá un input (nombre). El segundo paso también recogeremos información
por lo que  así que crearemos un formGroup (myForm2), tendrá tres inputs (nombre, nivel y probabilidad).

El mat-form-field requiere de controladores. El primer paso deberá tener un controlador, para el nombre (nombreColeccion).
En el segundo paso deberá tener un controlador, para el nombre (nombreCromo).

# CrearColeccion()

Hago el POST en la base de datos dandole un nombre la coleccion (y un logo).

Si he cargado un logo (imagenColeccionCargado = true), hacemos el POST en la base de datos del nombreImagenColeccion.
```

Figura 9.3 Componente Crear Colección

Este proyecto se compone de 47 componentes de diferentes módulos de la aplicación. como por ejemplo mis-grupos, crear-punto, mis-colecciones, equipos, etc. Para navegar entre las vistas se ha utilizado el servicio de Router proporcionado por Angular, como se observa en la **Figura 9.3.**

```
// GRUPOS
{ path: 'grupo/:id', component: GrupoComponent },
{ path: 'grupo/:id/editarGrupo', component: EditarGrupoComponent },

// GRUPOS --> PASAR LISTA
{ path: 'grupo/:id/pasarLista', component: PasarListaComponent },

// GRUPOS --> EQUIPOS
{ path: 'grupo/:id/equiposGrupo', component: EquiposComponent },
{ path: 'grupo/:id/equiposGrupo/editarEquipo', component: EditarEquipoComponent },


// GRUPOS --> JUEGOS
{ path: 'grupo/:id/juegos', component: JuegoComponent },
{ path: 'grupo/:id/juegos/juegoSeleccionado', component: JuegoSeleccionadoActivoComponent },
{ path: 'grupo/:id/juegos/juegoSeleccionadoInactivo', component: JuegoSeleccionadoInactivoComponent },
```

Figura 9.3 Servicio de Router

9.1.3. Servicios

Los servicios permiten obtener acceso de los datos del servidor sin necesidad de hacerlo desde el componente. Esta se separación se realiza con el fin de evitar el sobrecargo del componente y facilitar su comprensión. Una vez se obtienen los datos en el servidor, se suele acompañar de un post-procesamiento de los mismos y un manejo de errores entre otros. Además, al crear un servicio, se permite compartir la lógica de este entre los distintos componentes que lo requieran, mientras que al crearlo directamente en un componente sería exclusivo de una única vista.

En los servicios se incluyen todas las solicitudes realizadas a la base de datos mediante peticiones http, como métodos de obtención, modificación o eliminación de datos. En la **Figura 9.4** se muestra un ejemplo de un servicio que será llamado por un componente con el fin de realizar la función del mismo. En este proyecto se han creado un total de 10 servicios, como por ejemplo *alumnos*, *colección*, *juego-de-puntos*, etc.

```
export class AlumnoService {

  // URLs que utilizaremos
  private APIUrl = 'http://localhost:3000/api/Alumnos';
  private APIUrlGrupos = 'http://localhost:3000/api/Grupos';
  private APIUrlProfesor = 'http://localhost:3000/api/Profesores';

  private APIURLImagenAlumno = 'http://localhost:3000/api/imagenes/imagenAlumno';


  listaAlumnos: any = [];
  alumno: any;

  constructor(private http: HttpClient) { }

  // ASIGNAR ALUMNOS A UN PROFESOR
  POST_AlumnosAlProfesor(alumno: Alumno, profesorId: number): Observable<Alumno> {
    return this.http.post<Alumno>(this.APIUrlProfesor + '/' + profesorId + '/alumnos', alumno);
  }

  // NOS DEVUELVE LOS ALUMNOS DEL GRUPO CUYO IDENTIFICADOR PASAMOS COMO PARÁMETRO
  GET_AlumnosDelGrupo(grupoId: number): Observable<Alumno[]> {
    return this.http.get<Alumno[]>(this.APIUrlGrupos + '/' + grupoId + '/alumnos');
  }
```

Figura 9.4 Servicio de Alumno

9.2. Implementación Mobile

Para el desarrollo de la aplicación Mobile, se hace uso de gran parte de las clases, componentes y servicios de Dashboard, pero como ya se comentó anteriormente, Mobile es una versión simplificada de Dashboard por lo que no aparecerán todas las clases, ni servicios ni componentes.

9.2.1. Clases

Como ya se citó previamente, las clases permiten guardar/manejar la información perteneciente de la base de datos. Para la modalidad Mobile, se podrán encontrar las clases creadas en la carpeta ***src*** dentro de ***clases***. Los archivos tienen la extensión *TypeScript*.

Se han hecho uso 20 de 27 clases existentes en Dashboard.

9.2.2. Componentes y servicios

Los componentes en Mobile tendrán un papel muy importante ya que recopilarán tanto la interacción visual del usuario con el aplicativo cómo de los servicios utilizados para el correcto uso de las funcionalidades de Classpip.

Se debe realizar una distinción entre componentes cómo una página y componentes estéticos como el AccordionComponent.

Los componentes están formados por tres tipos de archivos:

- El controlador, también conocido como archivo TypeScript, el cual contiene los datos y la lógica del componente.
- La arquitectura e interfaz gráfica conocida como archivo HTML.
- Hoja de estilos gráficos conocida como archivo SCSS, el cual permite modificar la estética de botones, letras u otros componentes.

La modalidad Mobile se compone de 15 componentes de tipo páginas como puede ser el ejemplo de alumnos-equipo, alumnos-grupo además de 2 componentes de tipo estético como es el caso de AccordionGrupo y AccordionPuntos.

Como se ha citado anteriormente los Servicios en Mobile vienen introducidos en el propio componente de la página. Una futura mejora será el segregar los servicios en un único archivo y consecuentemente reducir la carga de código existente en el componente página.

Los Servicios utilizados son los mismos que Dashboard, pero particularizados para Ionic ya que Angular dispone de funciones que en Ionic son inexistentes.

En la **Figura 9.5** se muestras ejemplos de HTML y TS de un componente estético:

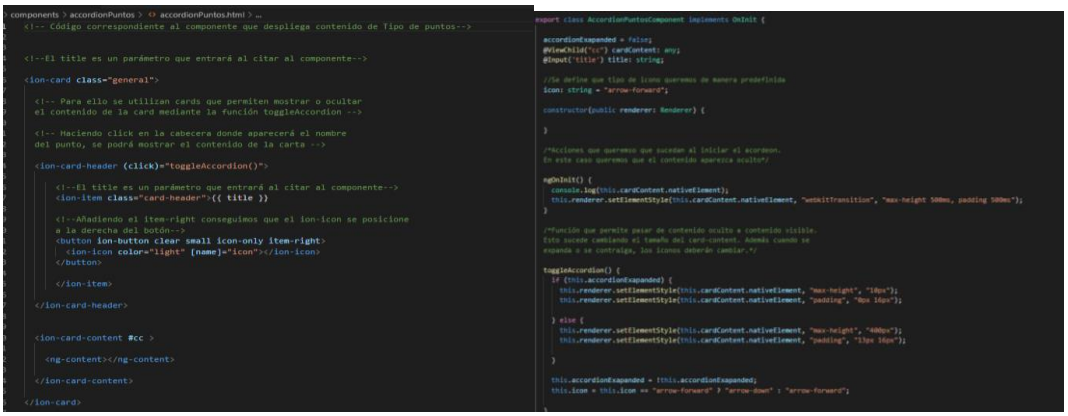


Figura 9.5 Ficheros HTML (izquierda) y TS (derecha)

De la misma manera se muestra en las **Figura 9.6** un ejemplo de archivo HTML y TS donde se puede apreciar que en un componente página se incluyen los servicios utilizados en la página.

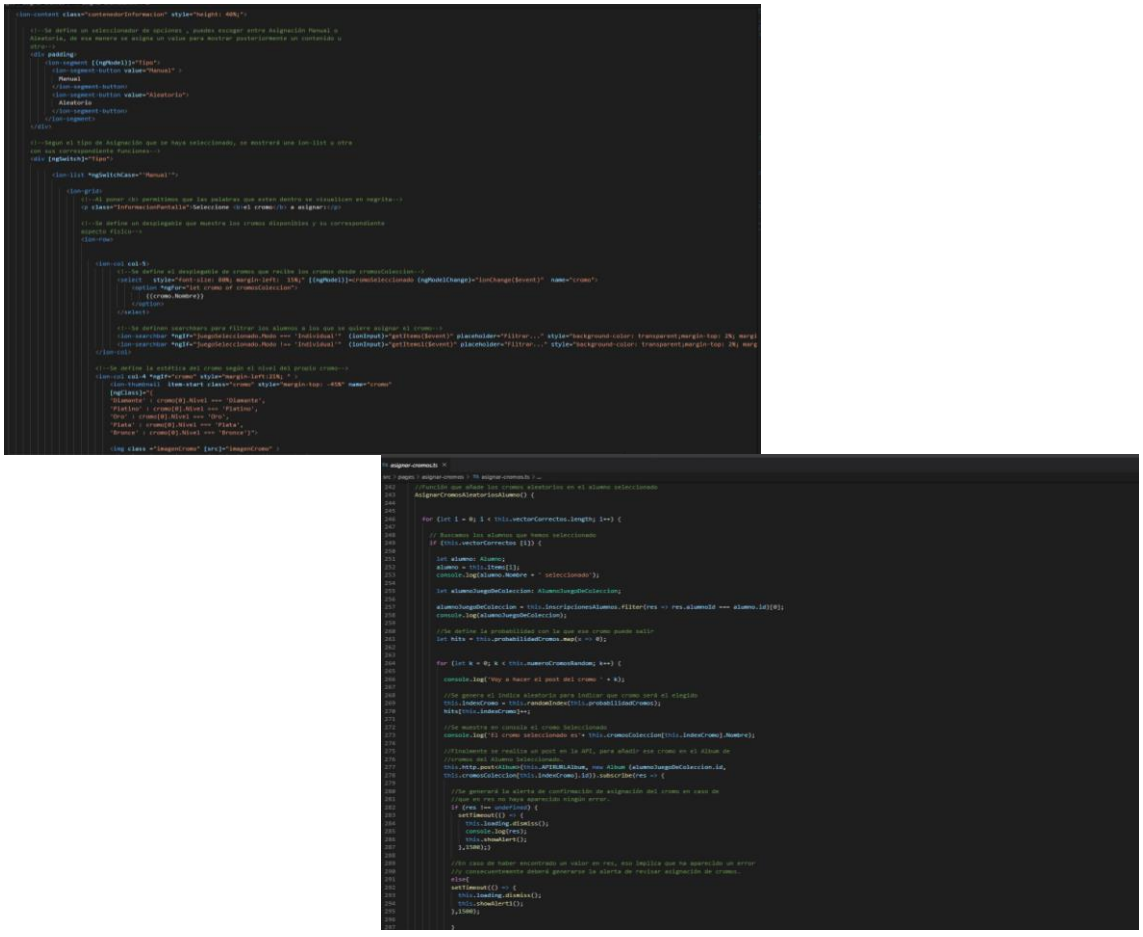


Figura 9.6 Ficheros HTML (arriba) y TS (debajo)

CAPÍTULO 10. PRUEBAS Y EVALUACIÓN

Una vez realizado la creación de los proyectos, se debe comprobar si todo funciona correctamente. Se han realizado unas pruebas que demuestran que en el momento en que se realizaron la aplicación funcionaba correctamente, a su vez las pruebas ayudan a los usuarios externos a explicar el funcionamiento de la misma.

Al desarrollar esta aplicación, se puede dar el caso que es más clara de lo que realmente es. Quizás carece de información relevante y el usuario externo se pierde en algún punto de la aplicación. Por lo que, es necesario realizar una evaluación externa para ver si es necesario realizar cambios en la aplicación.

10.1. Pruebas

Para ayudar a entender cómo funciona la aplicación y qué funcionalidades tiene, se han diseñado una serie de vídeos donde se muestran varias funciones que se pueden llevar a cabo:

- Grupos: https://youtu.be/aBreMzggPj4
- Juego de puntos: https://youtu.be/d2zrLGrAzk0
- Juego de colección: https://youtu.be/ipAMFaneFp0
- Colecciones: https://youtu.be/0s9TaeZDTME
- Puntos e insignias: https://youtu.be/FU1x8wvK83Y

Estos videos han ayudado a hacer las pruebas pertinentes para demostrar que la aplicación funciona correctamente.

En el tutorial de *Grupos* se crea un grupo y se añaden tres alumnos, una vez se finaliza la creación y se entra dentro del módulo Mis Grupos se observa que se ha creado correctamente, el nombre y la descripción coinciden y tiene los mismos tres alumnos que se han añadido anteriormente. Además, se muestra como se editar el grupo de manera satisfactoria cambiando el nombre y la descripción. También se muestra como la creación de los equipos funciona correctamente, eligiendo lo alumnos que tendrán los equipos y pudiéndolos traspasar de uno equipo a otro.

En el tutorial de *Juego de puntos* se muestra cómo se crea un juego de puntos y cómo se asignan los puntos a los alumnos de forma correcta, por lo que se determina con esta prueba que el apartado de juego de puntos funciona

correctamente. Lo mismo pasa con el tutorial de *Juego de colección* se observa la creación satisfactoria del juego y la asignación de cromos a los alumnos.

Tanto en el tutorial de *Colecciones* y *Puntos e insignias* se muestra cómo funciona la correctamente la creación y apartado de editar, para las colecciones se editan los cromos y añaden nuevos cromos. Para los puntos e insignias se edita el nombre y la descripción de forma que se confirma que en ambos módulos no existen fallos.

10.2. Evaluación

Al realizar la evaluación del aplicativo Classpip, se ha buscado abarcar la mayor diversidad de docentes existente. Para ello, se han evaluado a tres docentes de distintos países y distintos niveles educativos.

Una vez se seleccionaron los docentes, se realizó una videoconferencia en la cual inicialmente se les puso en contexto sobre que era Classpip y que objetivos quería cumplir. Posteriormente se permite al docente interactuar con el aplicativo Dashboard y Mobile y verificar si los objetivos citados inicialmente se cumplían.

A continuación, se citan las experiencias y comentarios recibidos por parte de los docentes:

Albert Martínez Saura – Profesor de Lengua Castellana en los Ángeles, Ciudad de California U.S.A

“Classpip es una herramienta que despierta la interacción de los alumnos con la educación mediante un sistema meritocrático, concienciando que el esfuerzo genera recompensas.”

Verónica Sánchez – Profesora de Educación Infantil en Escuela Antoni Botey, Badalona

“Classpip es una herramienta que permite involucrar a los más jóvenes con la educación sin que pierdan las ganas, motivándolos como si de un juego se tratase”

Ricardo Escribano Martín- Profesor de Soporte Intensivo de Escolarización Inclusiva en Escuela Montbui, Caldes de Montbui

“Con Classpip se abren las posibilidades de interacción entre docente y alumno, creando un espacio virtual donde construir e intercambiar conocimiento. Fomenta el trabajo cooperativo e inclusivo debido a su formato lúdico.”

Seguidamente se detallan los aspectos positivos y a mejorar del aplicativo Classpip:

Aspectos positivos:

- Amigable e intuitivo gráficamente.
- Robusto y el usuario se siente bien informado cuando se alerta de un error o una mala asignación.
- Versátil ya que permite definir dinámicas y mecánicas al gusto del docente.
- Disponer de un menú lateral en Mobile permite un acceso mucho más sencillo y rápido para el usuario, evitando retroceder pantallas hasta la deseada.
- El sistema de meritocracia resulta muy atractivo para los docentes ya que permite activar el interés de los alumnos en sus clases.
- Los procesos de creación de equipos, grupos resultan muy intuitivos ya que estos están restringidos por fases, evitando dudas en el usuario y evitando generar discordia entre datos.
- El generar el propio docente las colecciones de cromos permite segregar la clase según la motivación de cada alumno y consecuentemente saber cómo sacar el máximo rendimiento de ellos.

Aspectos por mejorar:

- Incorporación de cambio de idioma. Actualmente solo se encuentra disponible en Castellano.
- Poder realizar competiciones entre equipos de otros grupos. Pongamos el ejemplo de Ricardo, que quiere realizar una competición entre los mejores grupos de Matemáticas de tercero A y tercero B. Actualmente esta función no está activa ya que se restringe a competir entre equipos de un mismo grupo.
- En Dashboard sería necesaria un pequeño texto informativo sobre el potencial de la pantalla en la que se encuentra para así conseguir que el usuario sepa que puede conseguir en ella.
- A la hora de crear una colección, no permite pasar de la etapa 2 a la etapa de finalizar.

- En Mobile puede generar duda el tamaño de texto informativo de las pantallas, se propone utilizar pictogramas u otra pantalla en la cual te permita informarte sobre cómo funciona Classpip Mobile. Realizando Pictogramas podemos evitar excluir a gente con visibilidad reducida.

- Permitir definir roles a alumnos de un equipo y consecuentemente que opten a un tipo de punto u otro, de esta manera conseguimos crear equipos y que cooperen sabiendo el rol de cada uno.

CAPÍTULO 11. CONCLUSIONES

Las conclusiones han sido redactadas a nivel técnico y a nivel personal. Las conclusiones técnicas definen los resultados obtenidos a partir de los objetivos planteados al iniciar el proyecto y las conclusiones personales reflejan tanto los resultados obtenidos a nivel personal como las impresiones sobre el proyecto y el entorno de trabajo utilizado.

11.1. Conclusiones técnicas

El principal objetivo era restructurar el proyecto Dashboard, el Service y la web de Onboarding. Una vez se verificó el código heredado, se determinó que la complejidad de entender y restructurar todo el proyecto como se había diseñado era más elevada que realizar el proyecto desde cero, por lo que los objetivos cambiaron. Ahora nos enfrentábamos a un nuevo desafío, crear un Dashboard, Service y Mobile bien estructurado y de fácil comprensión, para permitir la fácil adaptación de nuevos programadores a este proyecto.

La primera mejora a gran escala fue la de desarrollar un Service desde cero, en el antiguo Services había ciertos modelos que no estaban bien y otros que eran difíciles de entender ya que no estaban bien explicados. Además, había ciertas relaciones que no estaban bien definidas como era la relación N:M. Resultaba más sencillo crear uno nuevo que ir modelo a modelo retocando los errores. Lo que más dificultades nos causo fue entender cómo funcionaba la relación N:M pero gracias a unos videos del tutor del proyecto, se consiguió encontrarle el sentido. La forma de almacenar las imágenes en la base de datos también se cambió realizándolo de forma más sencilla y ordenada.

Una vez se creó el Service, se comenzó con el desarrollo en paralelo del Dashboard y de Mobile. En cuanto a Dashboard unos de los principales objetivos era conseguir un código entendible y susceptible a cambios, todo esto ha quedado implementado de forma que todos los componentes tienen un *readme* que explica lo que hace cada función del mismo. Otro de los objetivos que se marcó fue conseguir que el estilo gráfico fuera centralizado y que existiera una cierta armonía entre módulos, ya que en el anterior Dashboard tenía el problema de que si cambiabas de módulo desaparecía la coherencia estética entre módulos.

En el proyecto Dashboard al ser realizado por dos integrantes del grupo simultáneamente se tuvo que utilizar la aplicación Git, un sistema de control de versiones, para realizar el mezclado de versiones. La organización de estas versiones fue la siguiente, de la rama Master, salía una rama que se llamaba BrancaPol, y otra rama que se llamaba BrancaAlbert. El procedimiento para no pisarse entre versiones era no modificar los mismos archivos, es decir, se debía intentar coordinar los componentes que iban a crear/editar cada integrante. Si cuando se realizaba el mezclado, ambos integrantes habían

editado los mismos archivos se debían de añadir los cambios de esos ficheros de forma manual. Gracias a Git hemos conseguido realizar un mezclado de versiones sin obtener errores.

En Mobile al utilizar Ionic causó ciertos problemas a la hora de importar librerías externas o de extensión Angular, por lo que se debían de generar manualmente componentes como el Accordion, no como en Dashboard que había la posibilidad de importar directamente librerías sin necesidad de crearlas manualmente. Otro de los objetivos cumplidos fue crear un Mobile Profesor funcional y que tuviera la mayoría de las características de Dashboard. Como se puede observar en la memoria se han conseguido crear tanto en Mobile Profesor como en Dashboard las funcionalidades explicadas en el Capítulo 4.

Se ha logrado cumplir con todos los principales objetivos definidos en este proyecto, estos objetivos eran:

- Crear un nuevo proyecto Services desde cero.

- Crear un nuevo proyecto Dashboard con los módulos ya existentes en las anteriores versiones. Creando un código lo más intuitivo y eficiente, para que los futuros programadores puedan trabajar más cómodamente.

- Crear un proyecto Mobile Profesor desde cero.

- Crear un estilo gráfico con coherencia entre módulos y que se centrará en una única página de estilos.

11.2. Propuestas futuras mejoras

De cara al futuro, este proyecto presenta amplias posibilidades. Se proponen las siguientes mejoras:

- Encontrar un nuevo uso del nivel y de la probabilidad de los cromos. Actualmente el nivel se relaciona con el color del cromo. Pero se podría relacionar la probabilidad con el color del cromo y así eliminar el concepto de nivel, simplificando los cromos.

- Una mejora que se podría aplicar en el juego de colecciones sería la de intercambiar entre alumnos cromos.

- Encontrar un nuevo uso para el concepto de insignias. Ya que su concepto es el mismo que los de los puntos, pero además se le añade una imagen.

- Mejorar la opción de añadir alumnos a un grupo masivamente, podría ser reseñada para que los nombres se introdujeran en forma de lista.

- Añadir la autentificación autentica para el inicio de sesión y crear un menú de configuración para que el profesor pueda editar sus datos.

- Corregir el problema del botón Finalizar en el módulo de Crea Colección, el código es el mismo que el utilizado en otros *steppers* pero este no funciona.

- Añadir la pantalla de Pasar Lista, guardando el registro de las fechas y las horas, de cada alumno del grupo.

11.3. Conclusiones personales

A nivel personal, cuando se nos propuso este proyecto sabíamos que nos enfrentamos un reto. Ya que, el tema escogido para este proyecto no tenía mucha relación con el Grado de Sistemas Aeroespaciales y, además, desconocíamos todas las tecnologías empleadas en él. Lo que nos hizo decantarnos por este proyecto fue el hecho de poder trabajar con algo totalmente distinto a lo que habíamos hecho anteriormente en el grado, un proyecto el cual implicaba a muchas personas y con el fin de poder sacarlo al mercado en un futuro. Aunque habíamos programado en otros lenguajes nos llamaba la atención aprender uno totalmente distinto y conseguir aprenderlo autónomamente, un trabajo que como ingenieros nos pasaría en el ámbito laboral.

Gracias a este proyecto hemos desarrollado algo fundamental que posiblemente no lo hubiéramos conseguido realizando otro tipo de proyectos más relacionados con la aeronáutica. Primeramente, nosotros vimos un problema en la sociedad, la adicción a los videojuegos, a esa adicción se le podía añadir la falta de interés por la educación. Pensando como ingenieros vimos una solución a estos dos problemas que fue desarrollando esta aplicación. Como experiencia ingenieril nos ha ayudado a saber planificar bien los tiempos, organización, y saber qué tipo de información debemos utilizar de internet sin tener la obligación de tener que generarla tu desde cero. Ya que los ingenieros no crean sino utilizan cosas ya existentes y las aplican a la situación oportuna.

Al principio, encontramos dificultades a la hora de introducirnos al nuevo el lenguaje y la forma en cómo se estructuraban los componentes y los servicios, algo que nunca antes habíamos visto. Fue un proceso de aprendizaje que nos llevó mucho tiempo asimilar, pero una vez fuimos capaces de entender cómo funcionaban las relaciones entre el Services y el Dashboard/Mobile, conseguimos ser lo más productivos posibles.

Mientras desarrollábamos el proyecto, nos han surgido varias tomas de decisiones las cuales han sido consensuadas por los tres integrantes del grupo. La primera de todas fue ponerse de acuerdo con las tareas de cada uno, una vez se alcanzó el acuerdo de cómo se iba a repartir las tareas, el proyecto empezó a darse forma. Además, se decidió crear un código de la forma más

robusta i clara posible, lo que nos has llevado más tiempo, pero permite cambios a medida que se vayan desarrollando las siguientes versiones del proyecto. Además, se ha comentado todo el código con el fin de facilitar los cambios en futuras versiones. Esto nos ha permitido crear una herramienta para la gamificación funcional y sencilla, que era el objetivo que nos habíamos propuesto una vez nos encontramos con un código el cual estaba mal estructurado y con dificultades a hacer cambios.

CAPÍTULO 12. ANEXOS

12.1. Manual de iniciación del proyecto

Cuando un alumno nuevo se inicia en el proyecto Classpip se encuentra un poco abrumado y perdido. Para ello se ha creado un material con tutoriales y manuales para que la entrada a este proyecto sea más fácil. Entre este material se incluye videos muy esclarecedores que recomendamos ver.
En este apartado se pretende explicar los pasos que realizamos nosotros cuando empezamos este proyecto.

12.1.1. Instalación del entorno

Cuando nos iniciamos en este proyecto, lo primero que tenemos que hacer es instalarnos todos los programas correspondientes. La explicación de la instalación se encuentra en la página web de Onboarding y es muy sencilla. Se puede instalar tanto para Windows, macOS y Linux siguiendo los pasos del siguiente link: http://classpip-onboarding.herokuapp.com/developer/install

12.1.2. Instalación de los proyectos

Una vez tenemos instalados todos los programas correctamente, es el momento de instalar los proyectos en vuestro equipo. Los proyectos están subidos a un repositorio de GitHub. GitHub no es más que un repositorio en la nube donde podemos ir guardando nuestro proyecto. Para entender completamente el funcionamiento de GitHub recomendamos que se visualice la Lección 0 de los videos de “Tutorial para herramientas de Classpip” que se puede encontrar en el canal de YouTube de nuestro tutor Miguel Valero.

12.1.3. Instalación del proyecto Services

Para instalar el proyecto deberemos seguir una serie de pasos muy sencillos:

1. Abrir un terminal.

2. Instalamos el strongloop mediante el comando **npm install -g strongloop**. Esto solo se utilizará para el proyecto Services, con lo que este paso solo se realizará una vez.

3. Escoger la ubicación donde queremos copiar la carpeta con el proyecto. Por ejemplo, imaginemos que la queremos crear en la carpeta “TFG” dentro de Documentos. Deberemos abrir la ubicación y copiar la ruta. Después en el terminal debemos poner cd seguido de la ruta donde queremos copiarlo. Esto se hace para abrir la ubicación. En el caso del ejemplo, quedaría así: **cd C:\Users\usuario\Documents\TFG**.

4. Una vez nos encontramos dentro de la carpeta, clonamos el repositorio. Esto estará particularizado para cada proyecto, en este caso, para el Services. Para esto simplemente debemos copiar lo siguiente: git clone https://github.com/ppa10/Services.git

5. Una vez se copia el repositorio, notaremos que en nuestra carpeta TFG se habrá creado otra carpeta llamada Services (el nombre del repositorio). Accederemos a ella mediante el comando **cd Services**. Una vez dentro de ella, ejecutamos el comando **npm install**.

6. Una vez acabada la instalación, probaremos si el proyecto funciona mediante el comando **node.** (se ha de poner el punto también). Si funciona correctamente nos saldrán dos enlaces de localhost:3000. Abrimos el que tiene explorer y visualizamos la API REST.

12.1.4. Instalación del proyecto Dashboard

1. Repetimos los pasos 1 y 3 de la instalación del proyecto Services.

2. Una vez nos encontramos dentro de la carpeta, clonamos el repositorio utilizando: git clone https://github.com/ppa10/Dashboard.git

3. Accedemos a la carpeta Dashboard mediante el comando **cd Dashboard**. Una vez dentro de ella, volvemos a ejecutar el comando **npm install**.

4. Una vez acabada la instalación, probaremos si el proyecto funciona mediante el comando **ng serve**. Tardará un poco en compilar, pero una vez acabado accederemos a la aplicación utilizando el link de localhost:4000.

12.1.5. Instalación del proyecto Mobile Profesor

1. Repetimos los pasos 1 y 3 de la instalación del proyecto Services.

2. Una vez nos encontramos dentro de la carpeta, clonamos el repositorio utilizando: git clone https://github.com/sergiiosp/ClasspipMobile.git

3. Accedemos a la carpeta Mobile mediante el comando **cd Mobile**. Una vez dentro de ella, volvemos a ejecutar el comando **npm install**.

4. Una vez acabada la instalación, probaremos si el proyecto funciona mediante el comando **Ionic serve --lab**. Tardará un poco en compilar, pero una vez acabado accederemos a la aplicación utilizando el link de localhost:8200. En caso de no tener instalado la extensión Lab, saltará una alerta informándonos sobre la necesidad de instalar la extensión para visualizar el proyecto en formato Mobile.

12.2. Modelos

A continuación, se muestran en tablas los modelos creados en el proyecto Service junto a sus propiedades y relaciones.

Tabla 12.1. Modelo de AlbumEquipo.

Propiedades	Relaciones	
Fecha	belongsTo	Cromo
	belongsTo	EquipoJuegoDeColeccion

Tabla 12.2. Modelo de Album.

Propiedades	Relaciones	
Fecha	belongsTo	Cromo
	belongsTo	AlumnoJuegoDeColeccion

Tabla 12.3. Modelo de AlumnoJuegoDeColeccion.

Propiedades	Relaciones	
-	belongsTo	Cromo
	belongsTo	EquipoJuegoDeColeccion
	hasMany (through Album)	Cromo

Tabla 12.4. Modelo de AlumnoJuegosDePuntos.

Propiedades	Relaciones	
PuntosTotalesAlumno	belongsTo	Alumno
	belongsTo	JuegoDePuntos
	hasMany (through HistorialPuntosAlumno)	Punto

Tabla 12.5. Modelo de Alumno.

Propiedades	Relaciones	
Nombre	hasMany (through Matricula)	Grupo
PrimerApellido	hasMany (through AsignacionEquipo)	Equipo
SegundoApellido	hasMany (through AlumnoJuegoDePuntos)	JuegoDePuntos
ImagenPerfil	hasMany (through AlumnoJuegoDeColeccion)	JuegoDeColeccion

Tabla 12.6. Modelo de AsignacionEquipo.

Propiedades	Relaciones	
-	belongsTo	Alumno
	belongsTo	Equipo

Tabla 12.7. Modelo de AsignacionPuntoJuego.

Propiedades	Relaciones	
-	belongsTo	JuegoDePuntos
	belongsTo	Punto

Tabla 12.8. Modelo de Coleccion.

Propiedades	Relaciones	
Nombre	hasMany	Cromo
ImagenColeccion	hasMany	JuegoDeColeccion

Tabla 12.9. Modelo de Cromo.

Propiedades	Relaciones	
Nombre	hasMany (through Album)	AlumnoJuegoDeColeccion
Imagen		
Probabilidad	hasMany (through AlbumEquipo)	EquipoJuegoDeColeccion
Nivel		

Tabla 12.10. Modelo de EquipoJuegoDeColeccion.

Propiedades	Relaciones	
-	belongsTo	Equipo
	belongsTo	JuegoDeColeccion
	hasMany (through AlbumEquipo)	Cromo

Tabla 12.11. Modelo de EquipoJuegoDePuntos.

Propiedades	Relaciones	
PuntosTotalesEquipo	belongsTo	Equipo
	belongsTo	JuegoDePuntos
	hasMany (through HistorialPuntosEquipo)	Punto

Tabla 12.12. Modelo de Equipo.

Propiedades	Relaciones	
Nombre	hasMany (through EquipoJuegoDePuntos)	JuegoDePuntos
	hasMany (through EquipoJuegoDePuntos)	JuegoDePuntos
FotoEquipo	hasMany (through EquipoJuegoDeColeccion)	JuegoDeColeccion

Tabla 12.13. Modelo de Grupo.

Propiedades	Relaciones	
Nombre	hasMany (through Matricula)	Alumno
	hasMany	Equipo
	hasMany	JuegoDePuntos
FotoEquipo	hasMany	JuegoDeColeccion
	hasMany	JuegoDeCompeticion
	hasMany	AsignacionEquipo

Tabla 12.14. Modelo de HistorialPuntosAlumno.

Propiedades	Relaciones	
ValorPunto	belongsTo	Punto
Fecha	belongsTo	AlumnoJuegoDePuntos

Tabla 12.15. Modelo de HistorialPuntosEquipo.

Propiedades	Relaciones	
ValorPunto	belongsTo	Punto
Fecha	belongsTo	EquipoJuegoDePuntos

Tabla 12.16. Modelo de Imagen.

Propiedades	Relaciones	
-	-	-

Tabla 12.17. Modelo de Insignia.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-
Imagen	-	-

Tabla 12.18. Modelo de Jornada.

Propiedades	Relaciones	
Nombre	-	-
Fecha	-	-

Tabla 12.19. Modelo de JuegoDeColeccion.

Propiedades	Relaciones	
-	hasMany (through AlumnoJuegoDeColeccion)	Alumno
	hasMany (through EquipoJuegoDeColeccion)	Equipo

Tabla 12.20. Modelo de JuegoDeCompeticion.

Propiedades	Relaciones	
TipoCompeticion	hasOne	Liga

Tabla 12.21. Modelo de JuegoDePuntos.

Propiedades	Relaciones	
-	hasMany	Nivel
	hasMany (through AsignacionPuntoJuego)	Punto
	hasMany (through AlumnoJuegoDePuntos)	Alumno
	hasMany (through EquipoJuegoDePuntos)	Equipo

Tabla 12.22. Modelo de Juego.

Propiedades	Relaciones	
Tipo	-	-
Modo	-	-
JuegoActivo	-	-

Tabla 12.23. Modelo de Liga.

Propiedades	Relaciones	
NumeroJornadas	hasMany	Jornada

Tabla 12.24. Modelo de Matricula.

Propiedades	Relaciones	
-	belongsTo	Alumno
	belongsTo	Grupo

Tabla 12.25. Modelo de Nivel.

Propiedades	Relaciones	
Nombre	hasMany	AlumnoJuegoDePuntos
PuntosAlcanzar		
PrivilegiosDelNivel	hasMany	EquipoJuegoDePuntos
Imagen		

Tabla 12.26. Modelo de Partido.

Propiedades	Relaciones	
JugadorUno	-	-
JugadorDos		
Ganador		

Tabla 12.27. Modelo de Profesor.

Propiedades	Relaciones	
Nombre	hasMany	Grupo
	hasMany	Alumno
Apellido	hasMany	Punto
	hasMany	ReglaManualColeccion
ImagenPerfil	hasMany	ReglaManualCompeticion
	hasMany	Coleccion
	hasMany	Insignia

Tabla 12.28. Modelo de Puntos.

Propiedades	Relaciones	
Nombre	hasMany (through AsignacionPuntoJuego)	JuegoDePuntos
	hasMany (through HistorialPuntosAlumno)	AlumnoJuegoDePuntos
Descripcion	hasMany (through HistorialPuntosEquipo)	EquipoJuegoDePuntos

Tabla 12.29. Modelo de ReglaAutomaticaColeccion.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-

Tabla 12.30. Modelo de ReglaAutomaticaCompeticion.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-

Tabla 12.31. Modelo de ReglaAutomaticaCompeticion.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-

Tabla 12.32. Modelo de ReglaManualColeccion.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-

Tabla 12.33. Modelo de ReglaManualCompeticion.

Propiedades	Relaciones	
Nombre	-	-
Descripcion	-	-